Transformers, Efficient Attention, and Enc / Dec Models

Lecture 4

18-789

Administrative

- The first student presentation extends to 2.3 (next Monday)
  - Please come to one of the instructor's office hour
- Check your AWS account
- Today
  - Pixel-RNN
  - Transformers
  - Efficient Attention
  - Enc / Dec Models

Recap – Autoregressive Models

Chain Rule Decomposition

$$p(x_1, \dots, x_T) = \prod_t p(x_t | x_1, \dots, x_{t-1})$$

Maximum Likelihood Estimation (MLE)

$$\underset{\theta}{\operatorname{argmax}} \sum_t \log p_\theta(x_t | x_{1,2,\dots,t-1})$$

Recap – Neural Architectures for AR

Chain Rule Decomposition

$$p(x_1, \dots, x_T) = \prod_t p(x_t | x_1, \dots, x_{t-1})$$

Maximum Likelihood Estimation (MLE)

$$\operatorname*{argmax}_\theta \sum_t \log p_\theta(x_t | x_{1,2,\dots,t-1})$$

Autoregressive Models	Recurrent	Masked	
	RNN/LSTM		
Parallel training	No		
Parameter sharing	Yes		

Recap – Neural Architectures for AR

Chain Rule Decomposition

$$p(x_1, \dots, x_T) = \prod_t p(x_t | x_1, \dots, x_{t-1})$$

Maximum Likelihood Estimation (MLE)

$$\operatorname*{argmax}_\theta \sum_t \log p_\theta(x_t | x_{1,2,\dots,t-1})$$

Autoregressive Models	Recurrent	Masked	
	RNN/LSTM	Masked MLP (MADE)	
Parallel training	No	Yes	
Parameter sharing	Yes	No	

Today – More Neural Architectures for AR

Chain Rule Decomposition

$$p(x_1, \dots, x_T) = \prod_t p(x_t | x_1, \dots, x_{t-1})$$

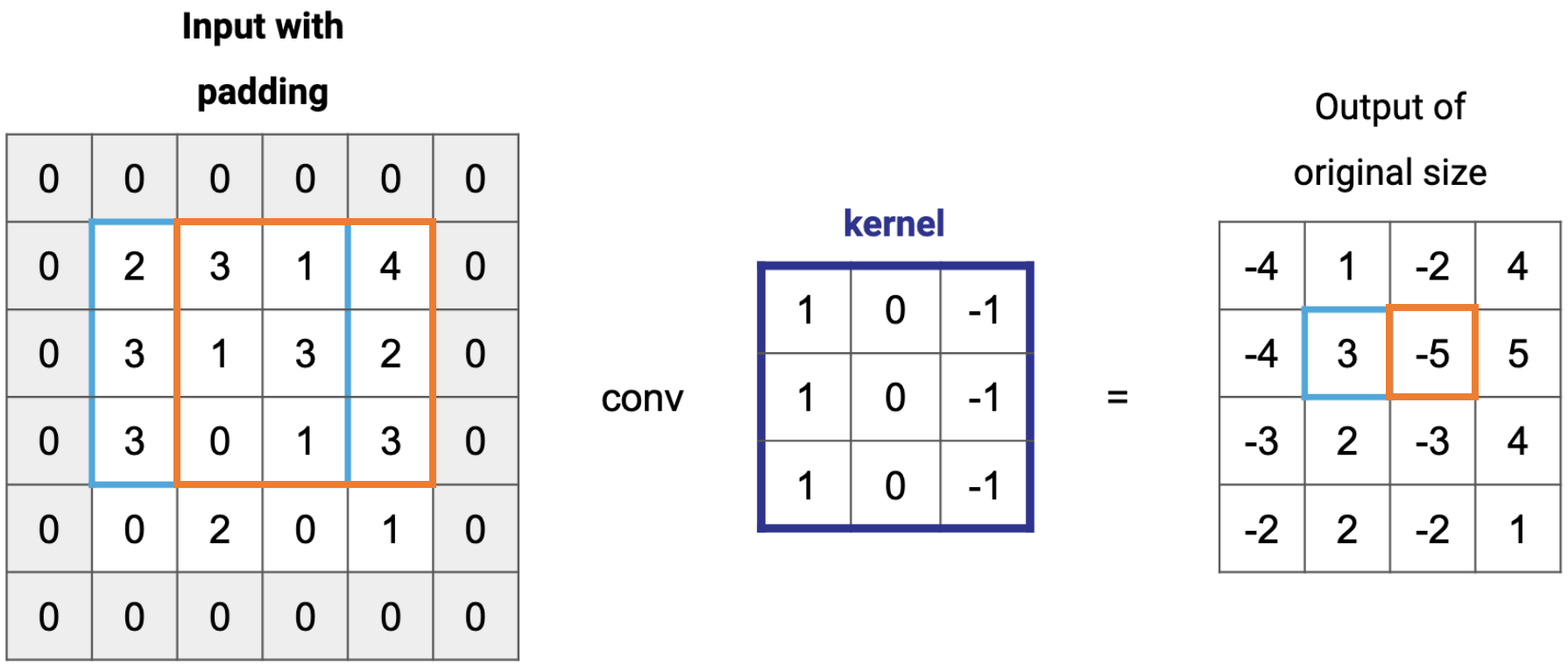
Maximum Likelihood Estimation (MLE)

$$\operatorname*{argmax}_\theta \sum_t \log p_\theta(x_t | x_{1,2,\dots,t-1})$$

Autoregressive Models	Recurrent	Masked	
	RNN/LSTM	Masked MLP (MADE)	**PixelCNN/Transformer/ Linear Attention**
Parallel training	No	Yes	Yes
Parameter sharing	Yes	No	Yes

Refresh: Convolution

- Convolution: slides a filter (kernel) over an input (e.g., image or other signals) to compute a weighted sum at each position.



Convolution: Translation Equivariance

- Good “inductive bias” for natural images

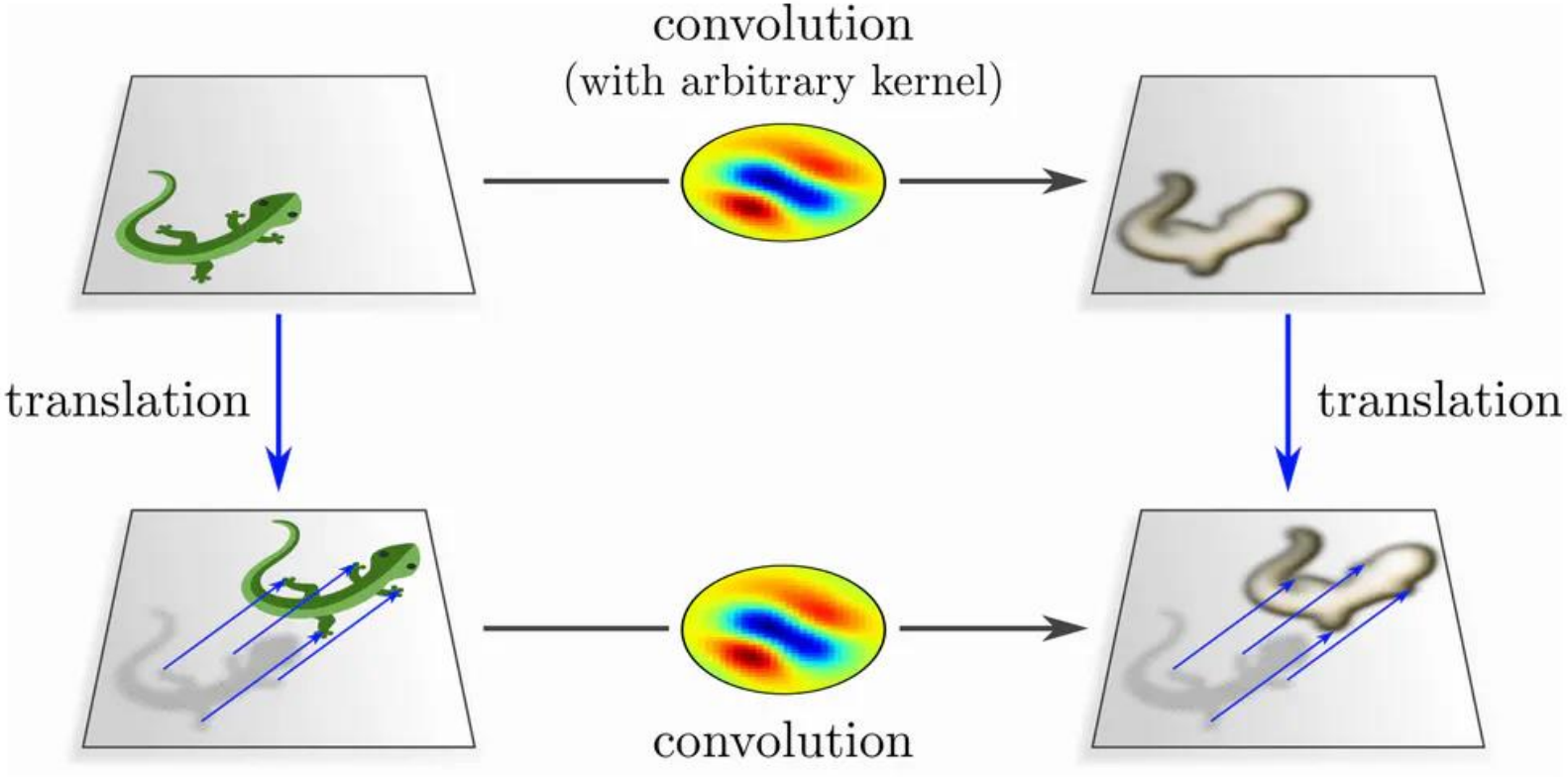
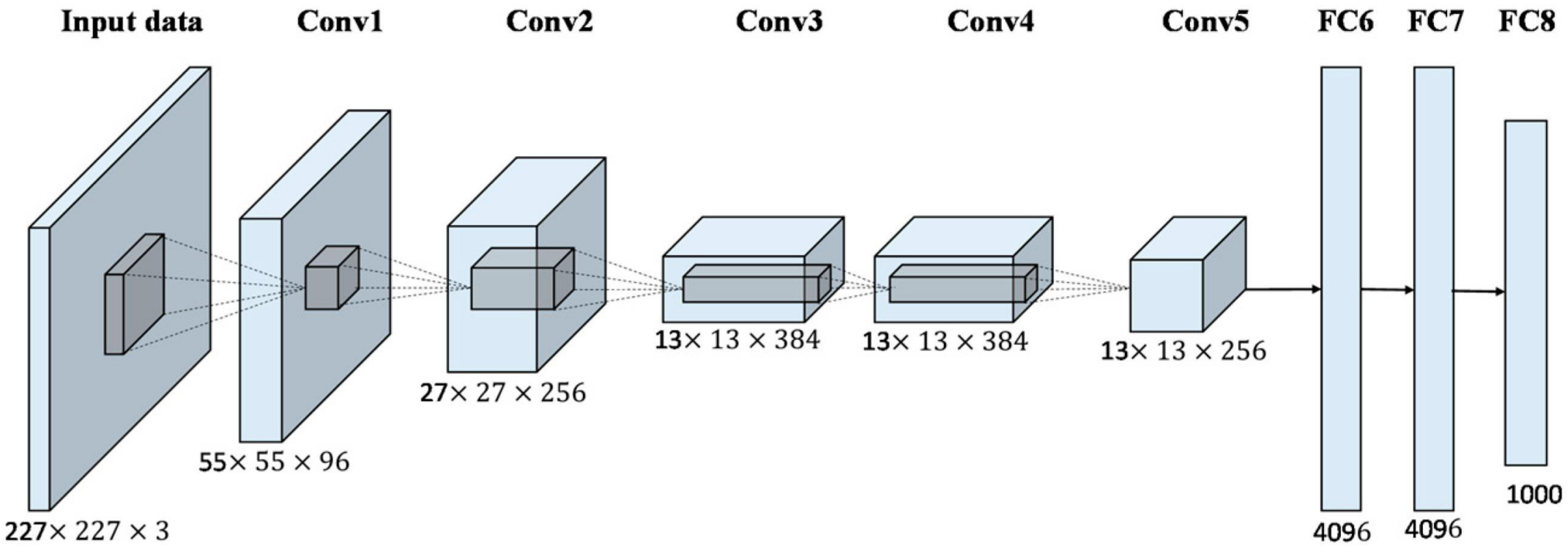


Figure from: https://maurice-weiler.gitlab.io/blog_post/cnn-book_2_conventional_cnns/

Convolutional Neural Networks (CNNs) for Image Classification



Krizhevsky et al., “ImageNet Classification with Deep Convolutional Neural Networks”, NeurIPS 2012

Receptive Field

- Receptive field: the size of the input region that affects a latent neuron
- Receptive field become bigger as we go deeper

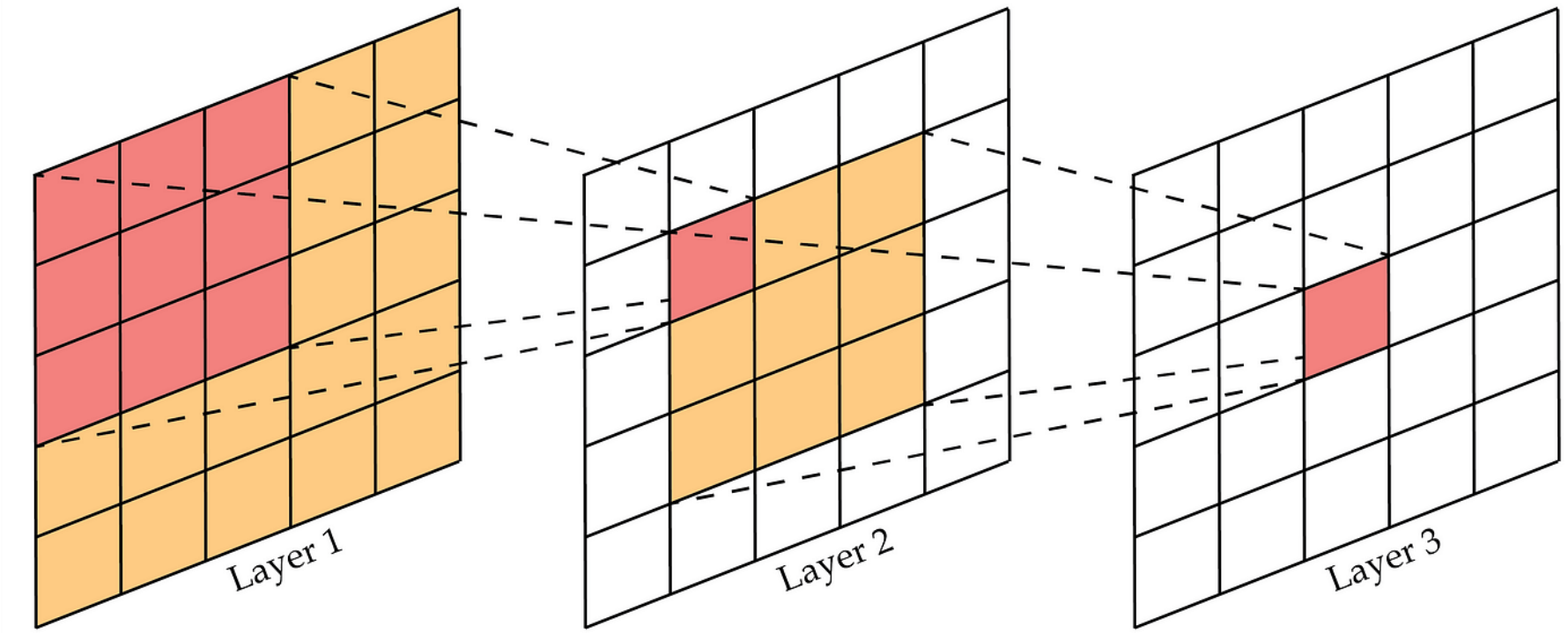
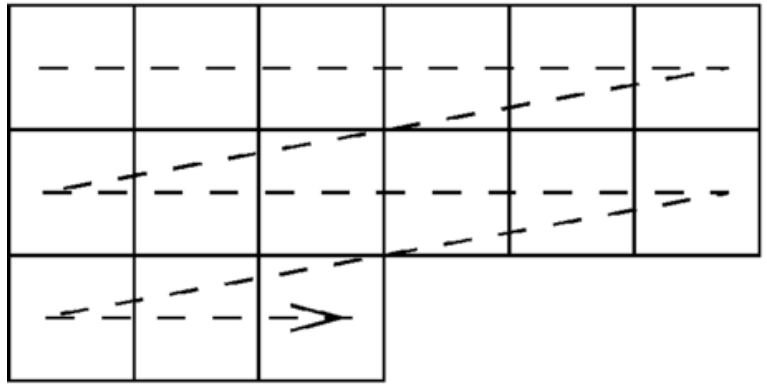


Figure from: https://medium.com/@rekalantar/receptive-fields-in-deep-convolutional-networks-43871d2ef2e9

PixelCNN: Convolution with Causal Masks

- We assume a “raster order” of pixels
- Convolution with “causal” mask



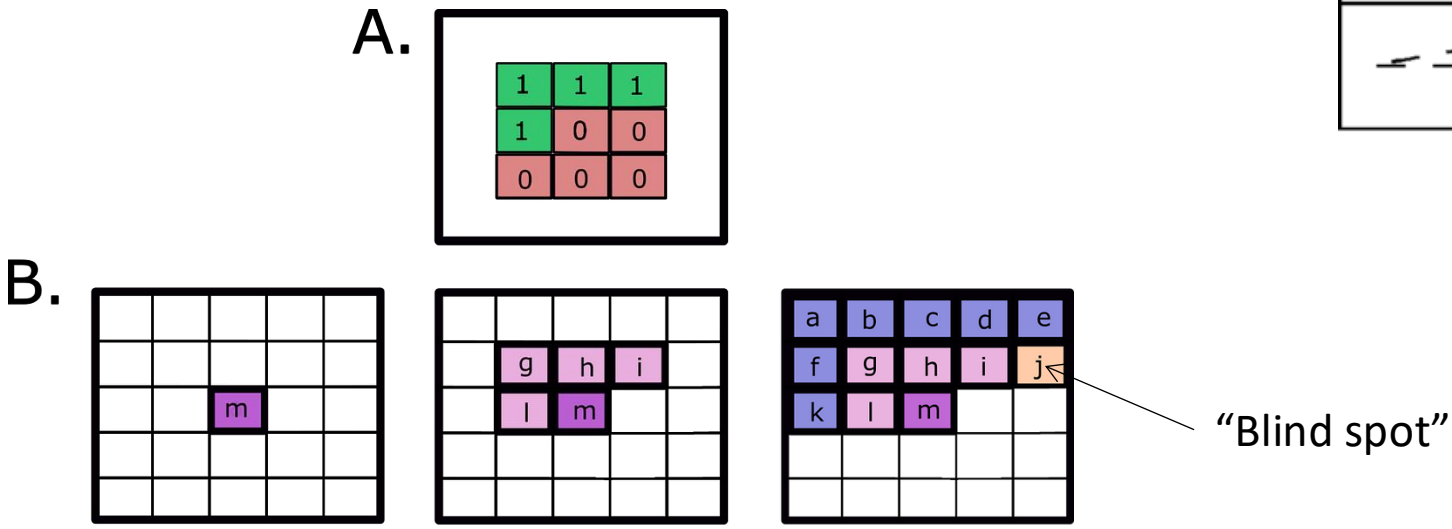
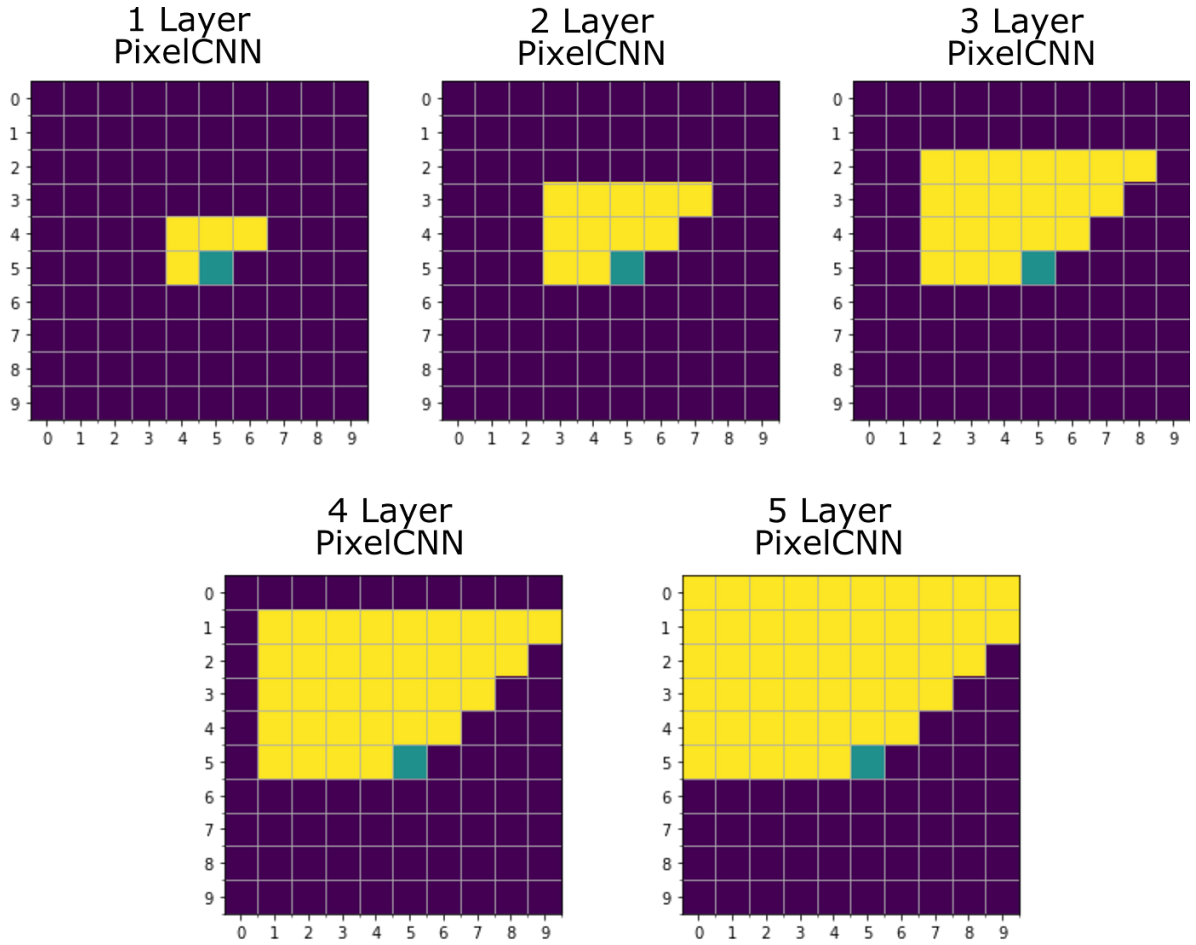


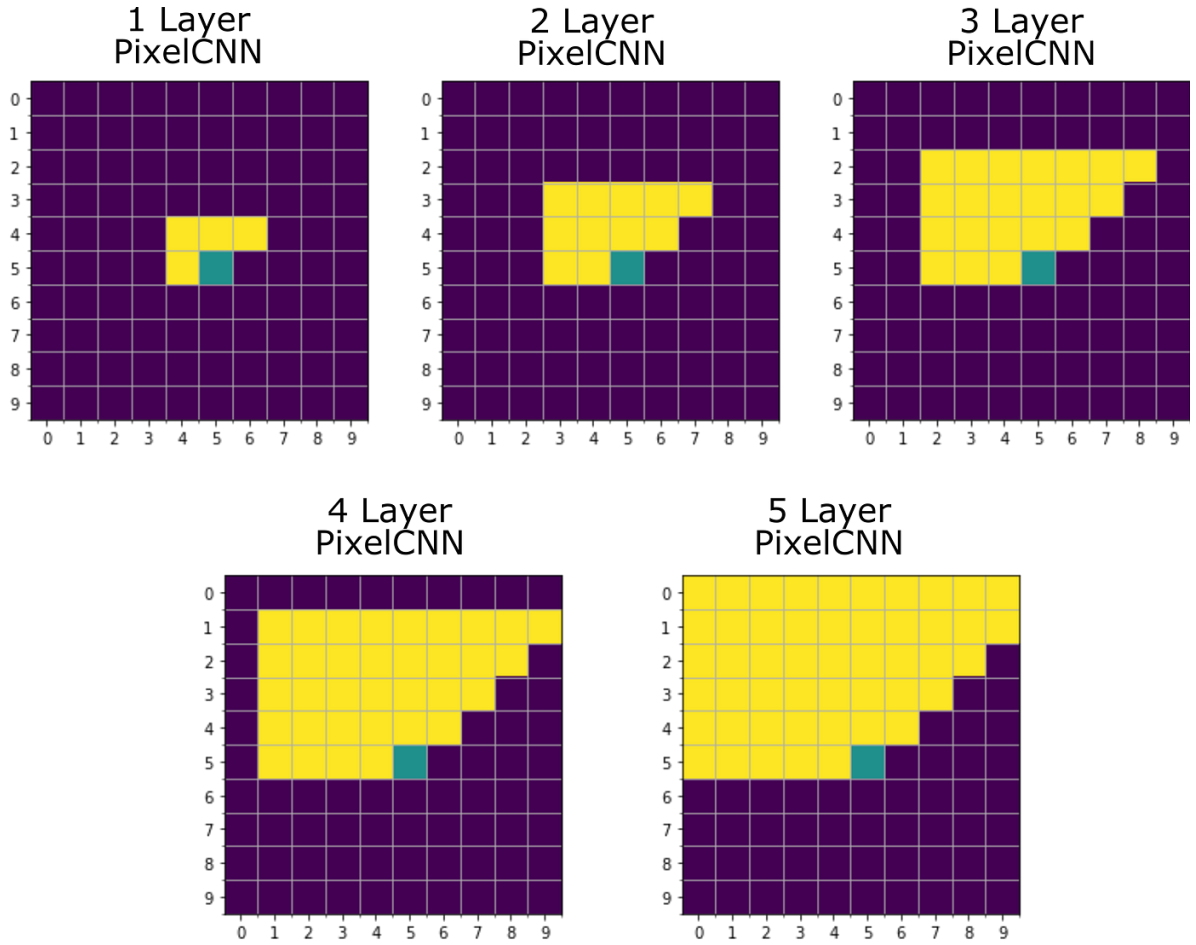
Figure from: https://towardsdatascience.com/pixelcnns-blind-spot-84e19a3797b9

PixelCNN: Convolution with Causal Masks



Receptive field after stacking multiple layers

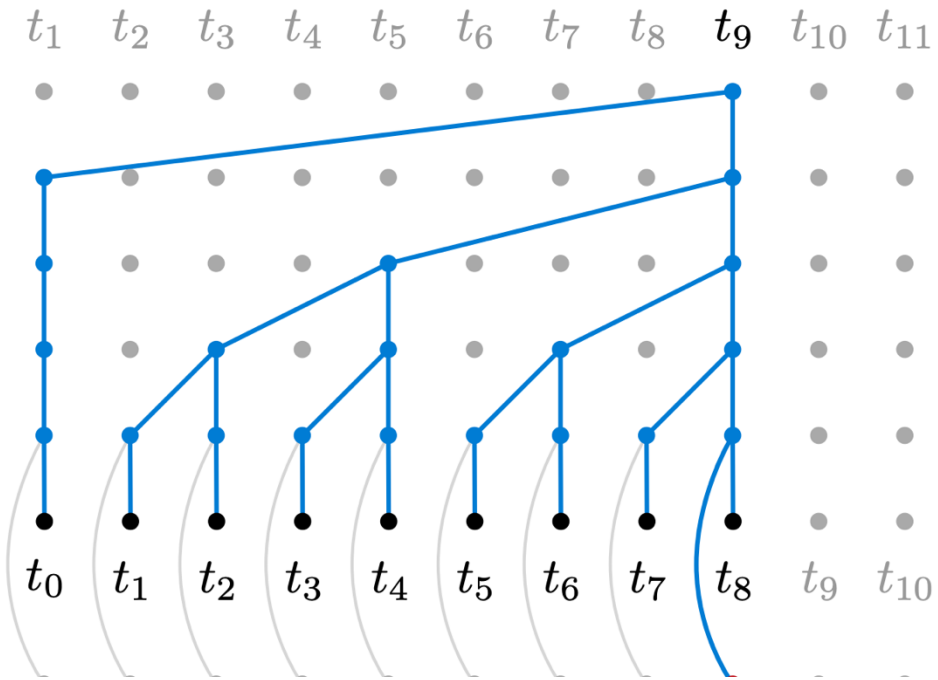
PixelCNN: Convolution with Causal Masks



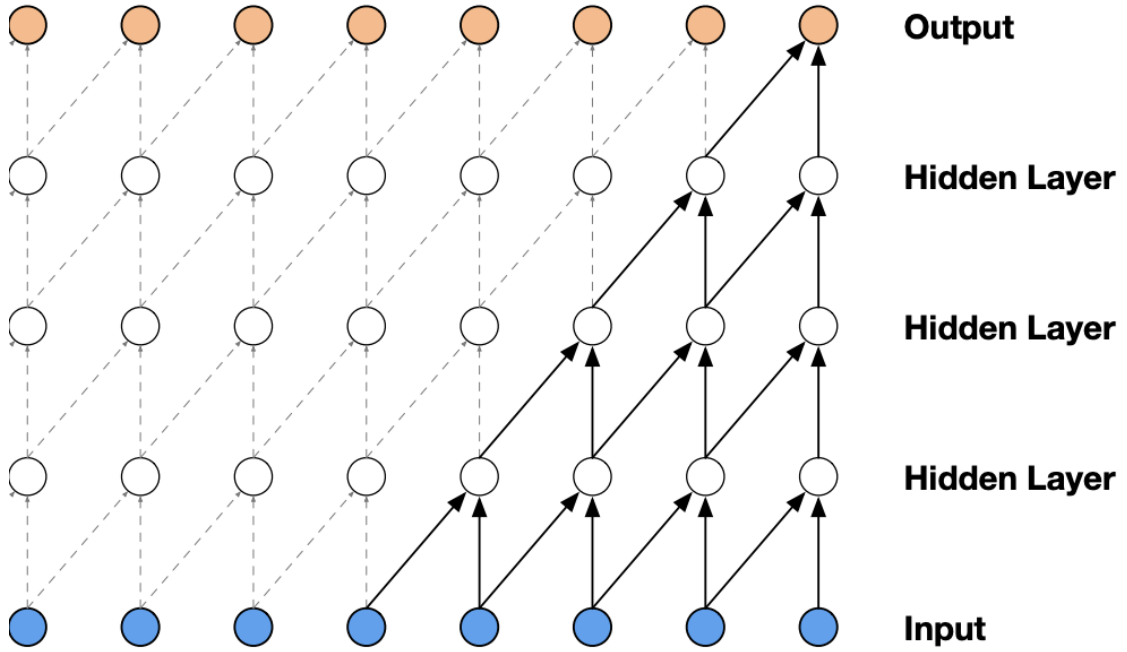
Receptive field after stacking multiple layers

Oord et al., "Conditional Image Generation with PixelCNN Decoders", NeurIPS 2016

1D Causal CNN for Sequential Data



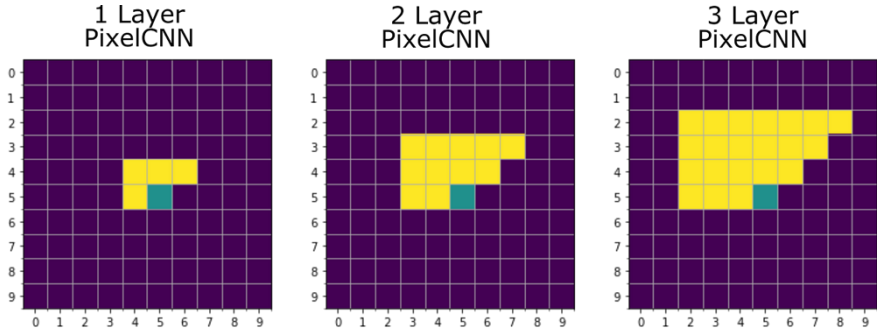
ByteNet, 2017

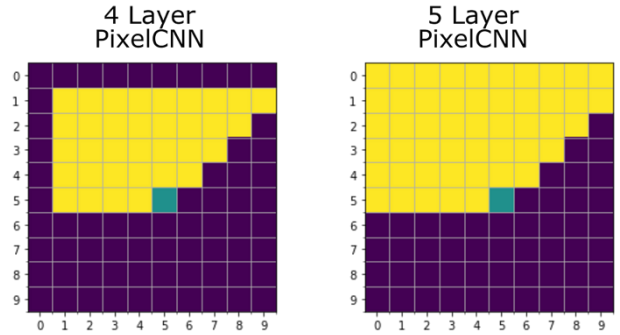


WaveNet, 2016

CNN cannot handle long-range dependencies

- receptive_field = kernel_size + num_layers * (kernel_size – 1)
- Larger kernel/more layers -> computationally more expensive
- “Dilation” can help a little bit
  - WaveNet: A Generative Model for Raw Audio





- Transformer/self-attention to the rescue!

Masked Attention

A recurring problem for convolution:

- limited receptive field → hard to capture long-range dependencies

(Self-) Attention: an alternative that has

- Unlimited receptive field (vs. Pixel-CNN)
- Also O(1) parameter scaling w.r.t data dimension (vs. MADE)
- Parallelized computation (vs. RNN)

Transformer Architecture

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

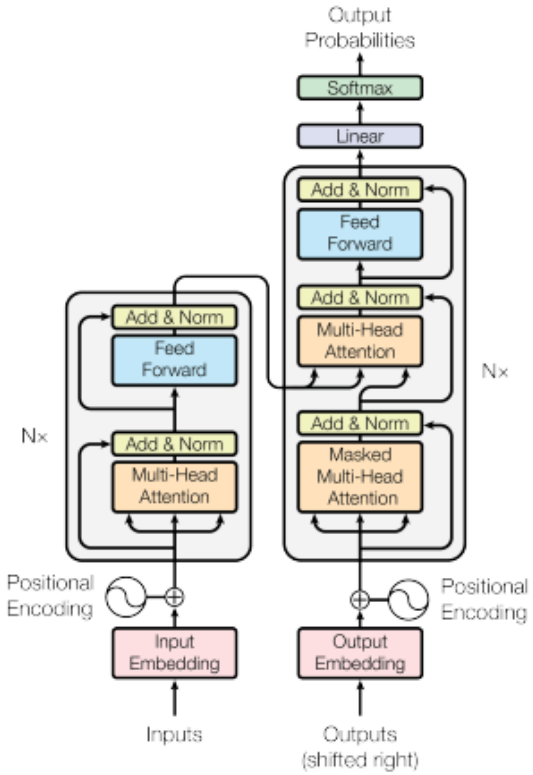


Figure 1: The Transformer - model architecture.

Attention

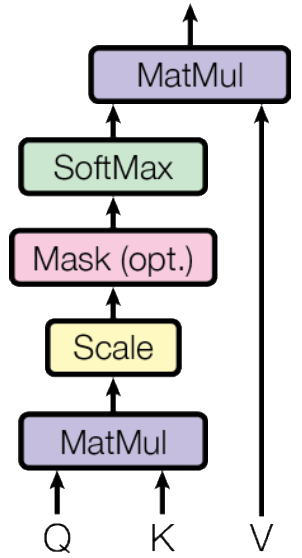
A sequence of <key, value> embeddings pairs

- The values are always the hidden states from a previous layer of the neural network. The attention mechanism outputs a weighted sum of these.
- For encoder-decoder attention, the values are the final hidden states of the encoder (as we so in the previous slide) and the keys are the hidden states from the target sequence.

A sequence of query embeddings

- The query is the current focus of the attention.
- We choose weights for each of the values by computing a score between the current query and each of the keys.

Scaled Dot-Product Attention



Attention output at position j =

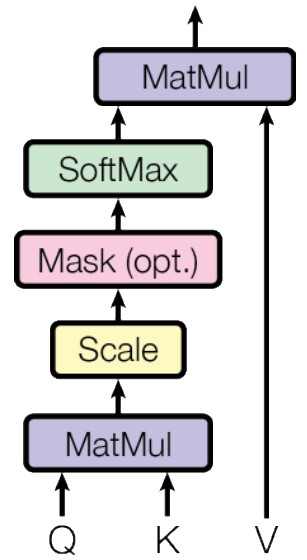
Score(q_j, k_i) =

Attention

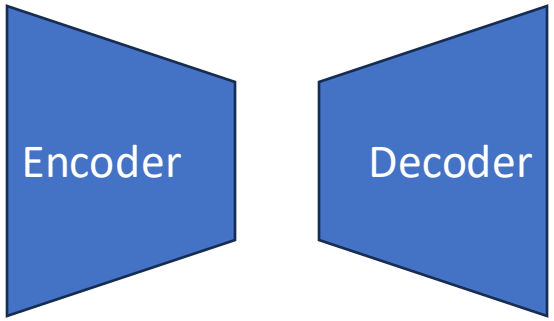
Since the attention computations at each position j are completely independent, we can actually parallelize all these computations and instead think in terms of matrix multiplications.

$$\text{Attention}(Q, K, V) =$$

Scaled Dot-Product Attention



Encoder-Decoder



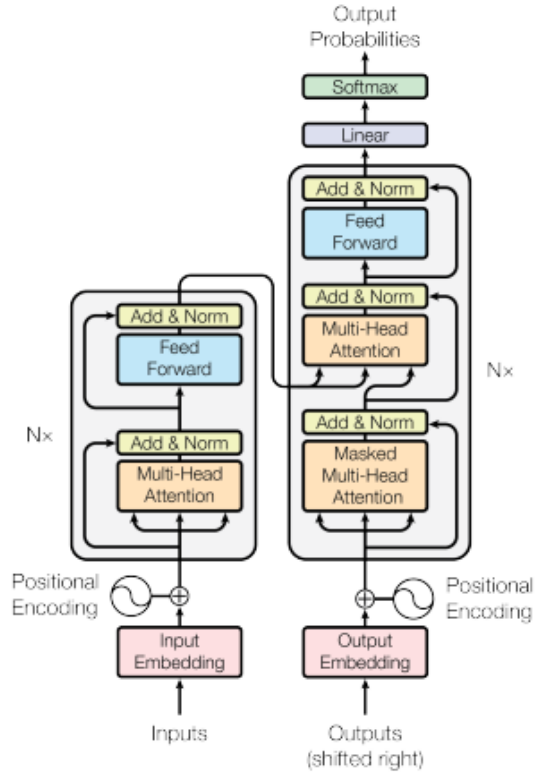
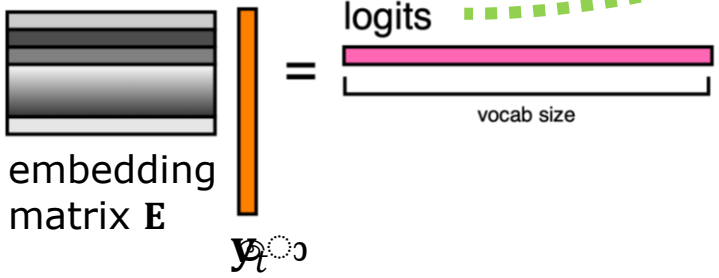


Figure 1: The Transformer - model architecture.

Transformer Output



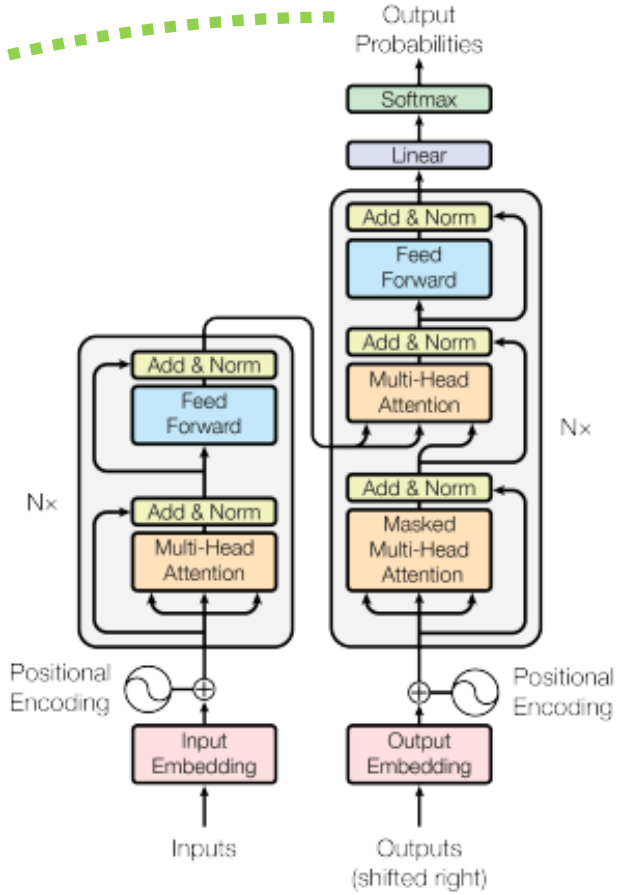
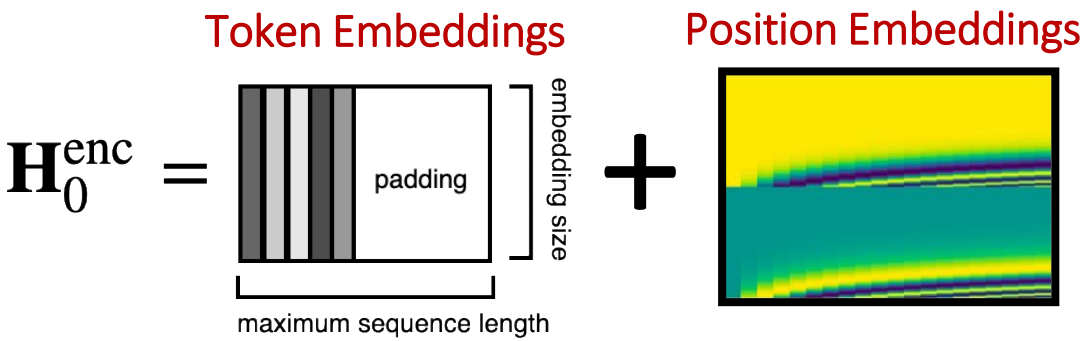


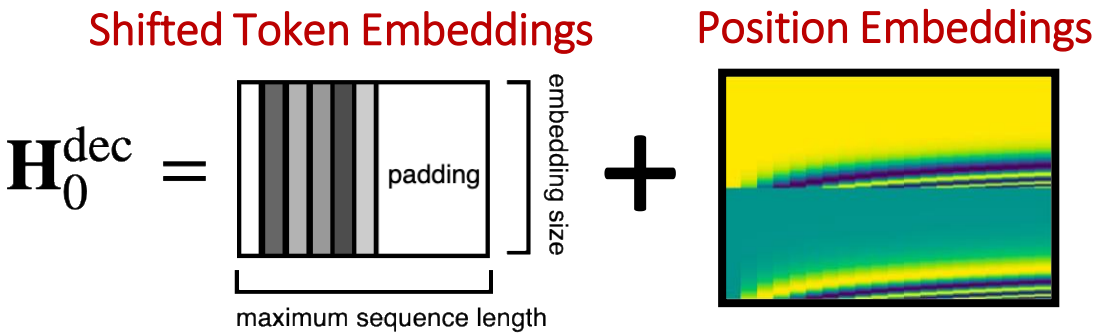
Figure 1: The Transformer - model architecture.

$P(y_t = i \mid x_{<T}, y_{<t}) =$

Transformer Input



The input to the decoder looks like:



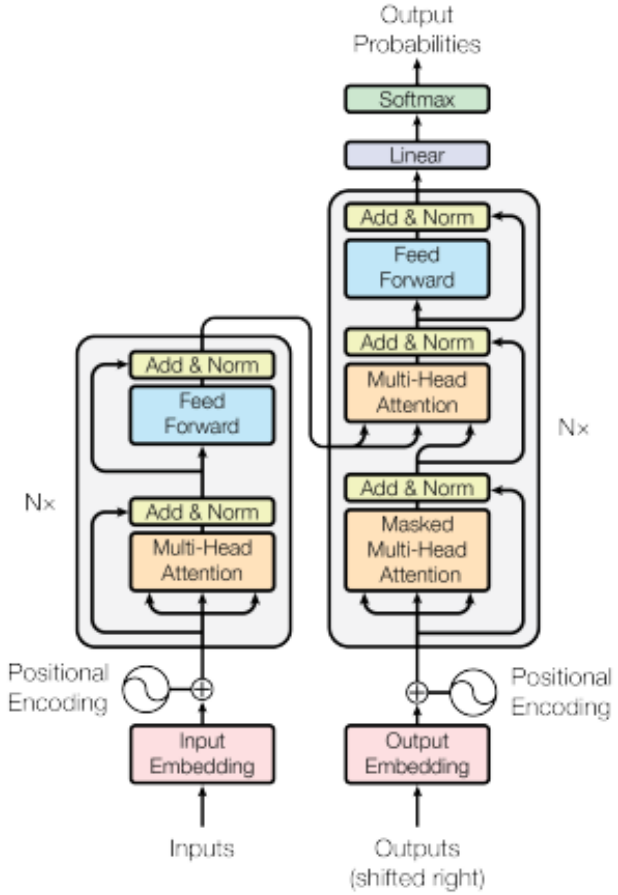


Figure 1: The Transformer - model architecture.

Positional Encoding

- Language is not permutation invariant
  - (“The mouse ate the cat” vs “The cat ate the mouse”)
- Need to encode position of each word
  - just add something

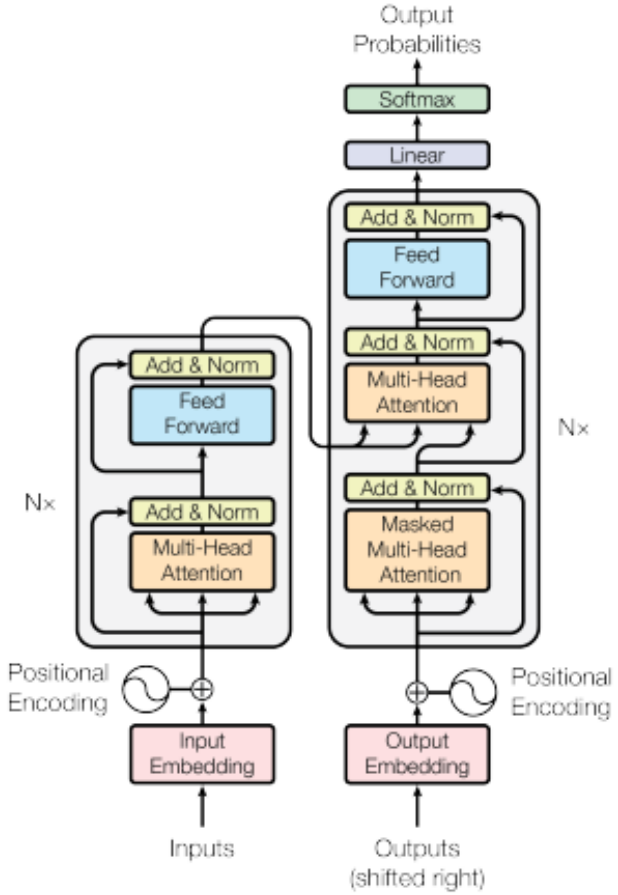
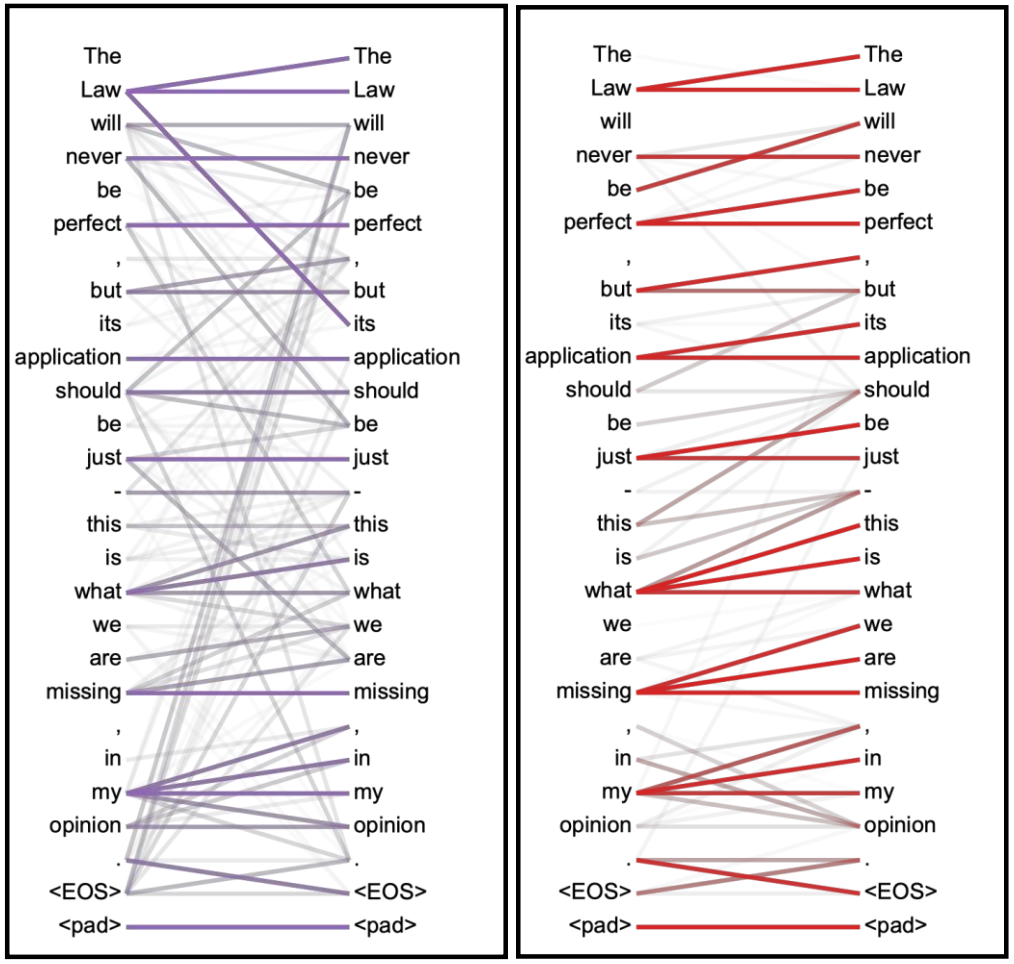


Figure 1: The Transformer - model architecture.

Self-Attention



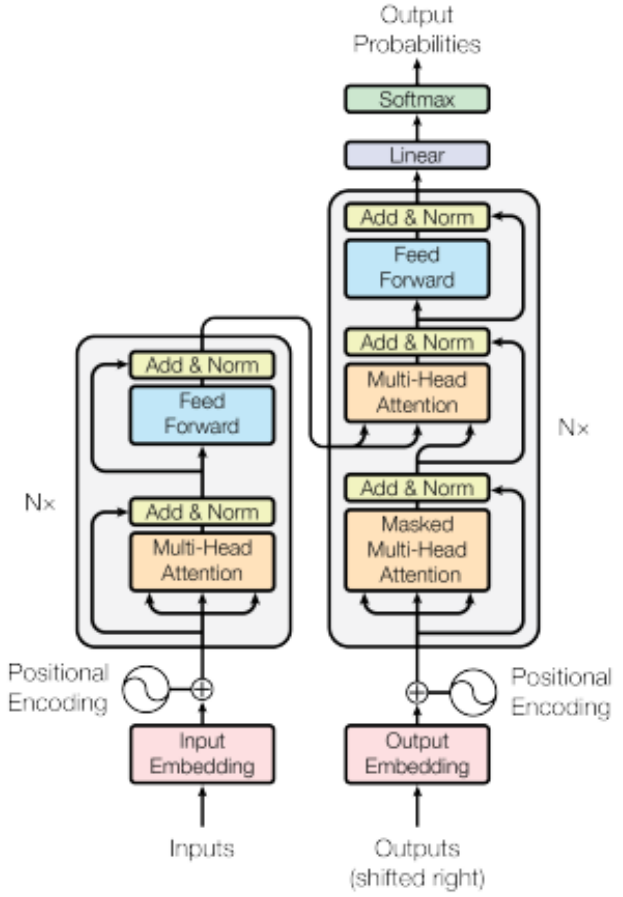
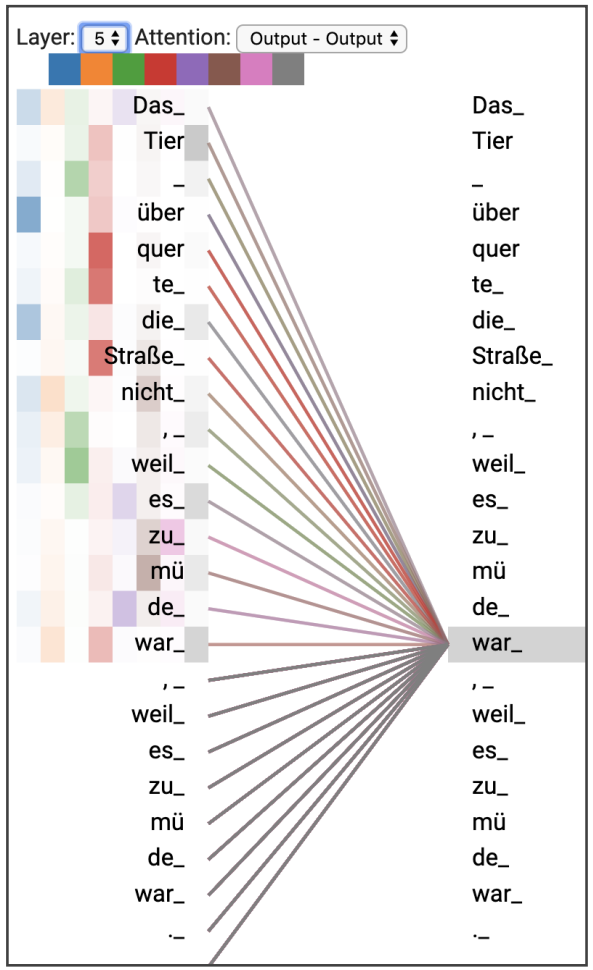


Figure 1: The Transformer - model architecture.

Cross-Attention



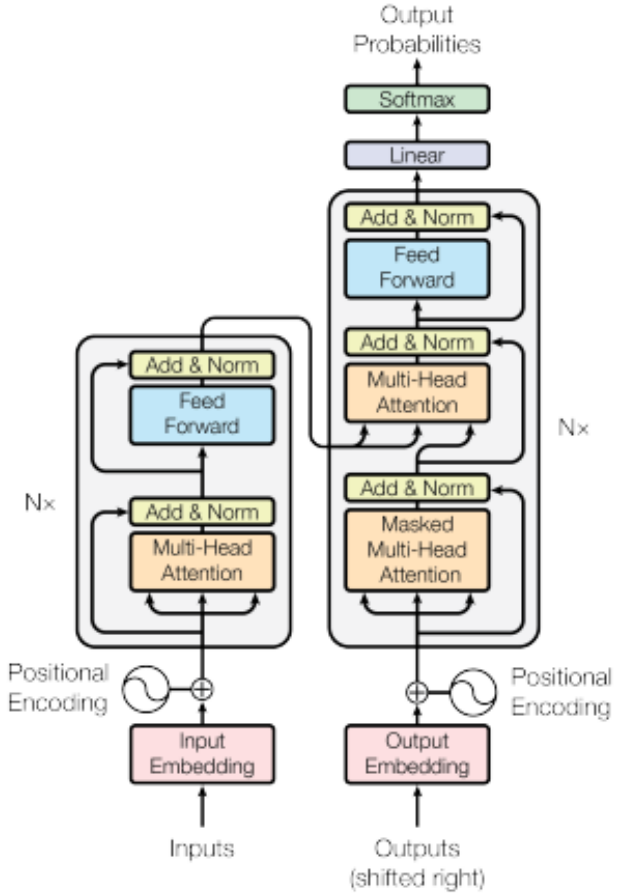


Figure 1: The Transformer - model architecture.

Layer Norm and Residual

- Normalization also dramatically improves trainability. There‘s post-norm (original) and and pre-norm (modern).

- Each module's output has the exact same shape as its input. Following ResNets, the module computes a "residual" instead of a new value:

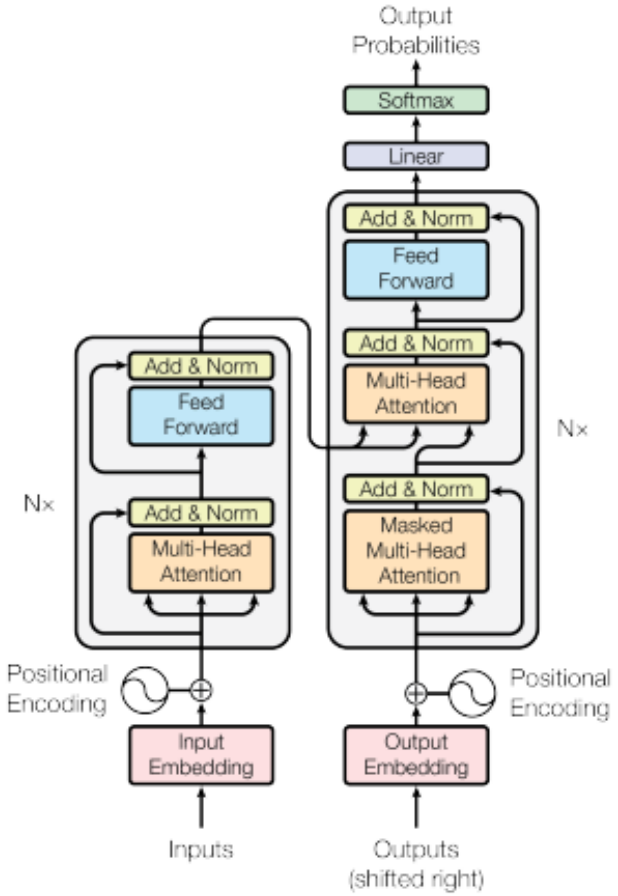


Figure 1: The Transformer - model architecture.

Encoder Step by Step

- MHA
- Add & Norm
- Feed Forward
- Add & Norm

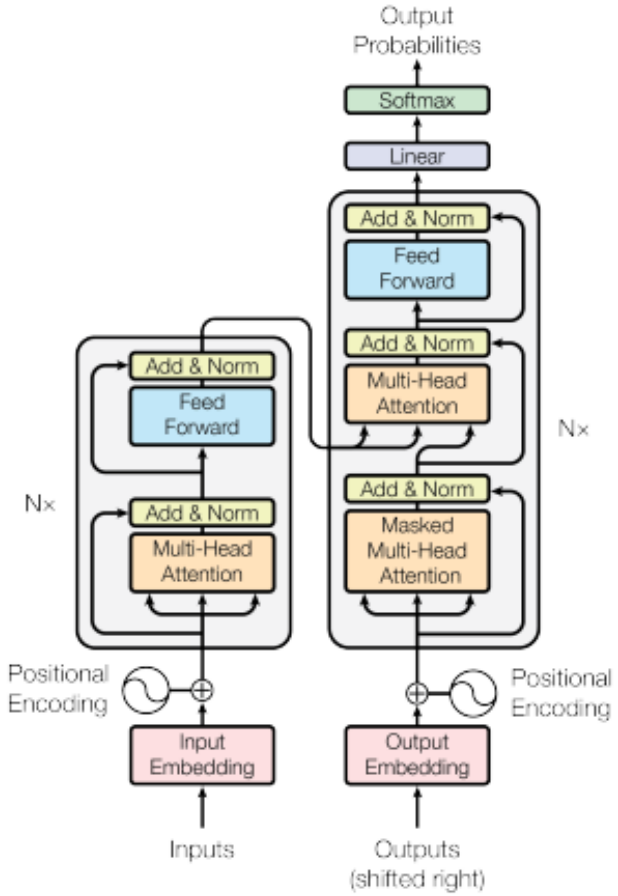


Figure 1: The Transformer - model architecture.

Decoder Step by Step

- Masked MHA
- Add & Norm
- Enc-Dec MHA
- Add & Norm
- Feedforward
- Add & Norm

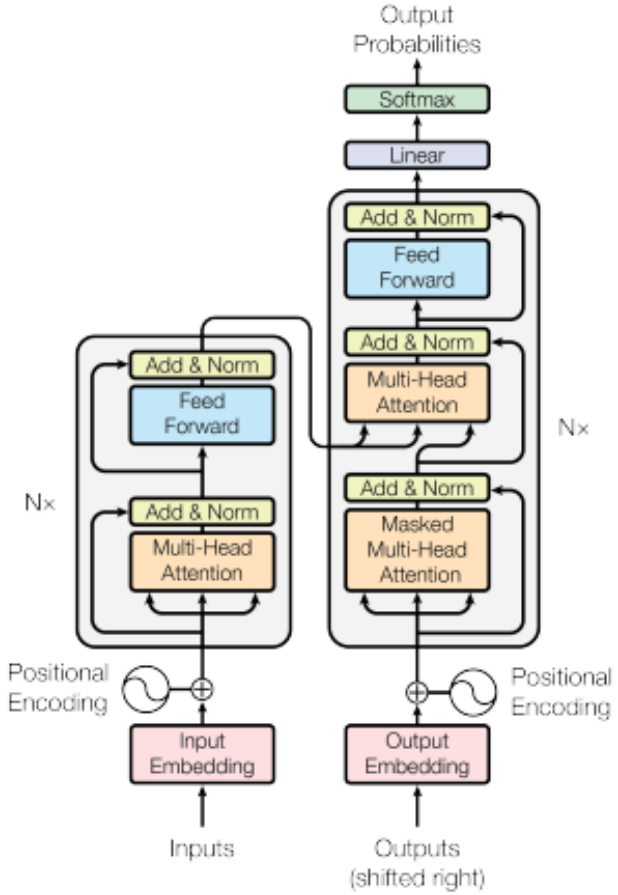


Figure 1: The Transformer - model architecture.

Attention is All You Need

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

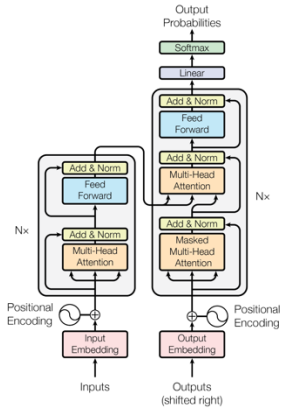
Animation source: https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Attention

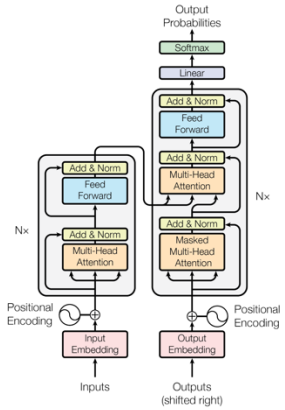
- RNNs:
  - (+) LSTMs work reasonably well for long sequences.
  - (-) Expects an ordered sequences of inputs
  - (-) Sequential computation: subsequent hidden states can only be computed after the previous ones are done.
- Transformer:
  - (+) Good at long sequences. Each attention calculation looks at all inputs.
  - (+) Can operate over unordered sets or ordered sequences with positional encodings.
  - (+) Parallel computation: All alignment and attention scores for all inputs can be done in parallel.
  - (-) Requires a lot of memory: N x M alignment and attention scalers need to be calculated and stored for a single self-attention head. (but GPUs are getting bigger and better)

Why Should I Know Transformer

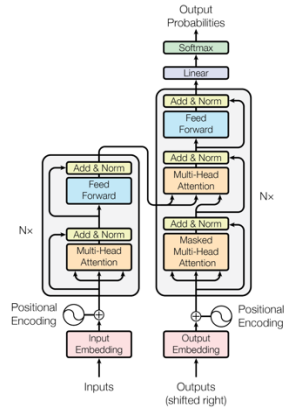
Computer Vision



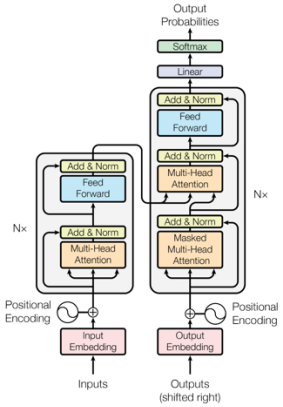
NLP



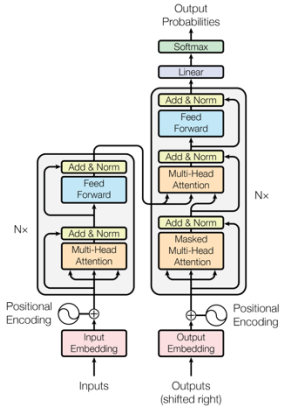
Reinforcement Learning



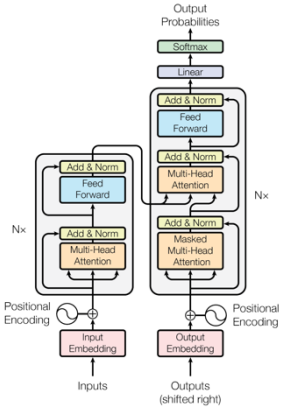
Speech



Translation



Graph/Science

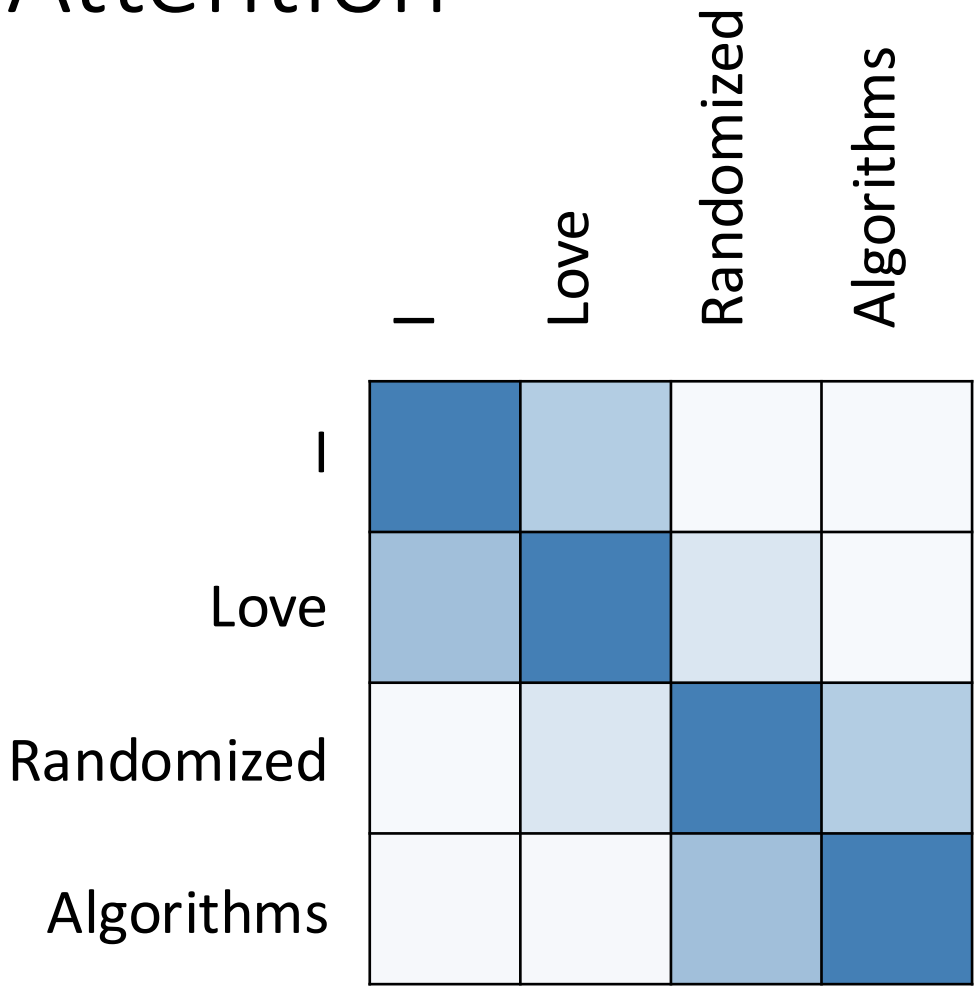


*Lucas Beyer

Transformer Problems

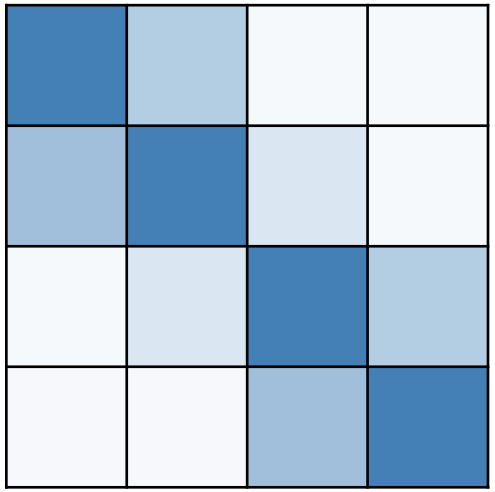
- Attention is quadratic during training
- Attention causes memory bottleneck during inference
- Positional Encoding extrapolation failed

Efficient Attention

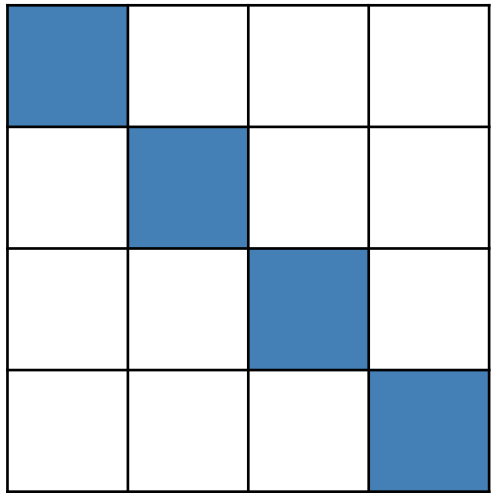


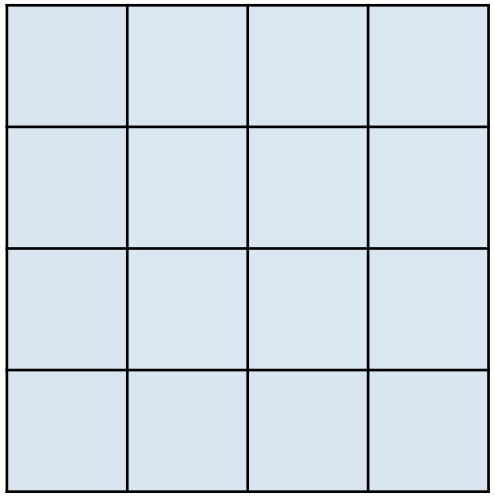
Attention Matrix Approximation

Sparse Approximation
(Child et al. 19,
Kitaev et al. 20)



Low-rank
(Katharopoulos et al. 20,
(Choromanski et al. 20)





Linear Transformer

Transformers are RNNs:
Fast Autoregressive Transformers with Linear Attention

Angelos Katharopoulos [1 2] **Apoorv Vyas** [1 2] **Nikolaos Pappas** [3] **François Fleuret** [2 4 *]

Abstract

Transformers achieve remarkable performance in several tasks but due to their quadratic complexity, with respect to the input's length, they are prohibitively slow for very long sequences. To address this limitation, we express the self-attention as a linear dot-product of kernel feature maps and make use of the associativity property of matrix products to reduce the complexity from $\mathcal{O}\left(N^2\right)$ to $\mathcal{O}\left(N\right)$, where N is the sequence length. We show that this formulation permits an iterative implementation that dramatically accelerates autoregressive transformers and reveals their relationship to recurrent neural networks. Our *linear transformers* achieve similar performance to vanilla transformers and they are up to 4000x faster on autoregressive prediction of very long sequences.

by the global receptive field of self-attention, which processes contexts of N inputs with a quadratic memory and time complexity $\mathcal{O}\left(N^2\right)$. As a result, in practice transformers are slow to train and their context is *limited*. This disrupts temporal coherence and hinders the capturing of long-term dependencies. Dai et al. (2019) addressed the latter by attending to memories from previous contexts albeit at the expense of computational efficiency.

Lately, researchers shifted their attention to approaches that increase the context length without sacrificing efficiency. Towards this end, Child et al. (2019) introduced sparse factorizations of the attention matrix to reduce the self-attention complexity to $\mathcal{O}\left(N\sqrt{N}\right)$. Kitaev et al. (2020) further reduced the complexity to $\mathcal{O}\left(N \log N\right)$ using locality-sensitive hashing. This made scaling to long sequences possible. Even though the aforementioned models can be efficiently trained on large sequences, they do not speed-up autoregressive inference.

$$A_l(x) = V' = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V.$$

$$V'_i = \frac{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right)}.$$

$$V'_i = \frac{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right) V_j}{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right)},$$

$$V'_i = \frac{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right) V_j^T}{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right)}.$$

$$\phi\left(x\right) = \text{elu}(x) + 1,$$

$$\begin{aligned}
s_0 &= 0, \\
z_0 &= 0, \\
s_i &= s_{i-1} + \phi\left(x_i W_K\right)\left(x_i W_V\right)^T, \\
z_i &= z_{i-1} + \phi\left(x_i W_K\right), \\
y_i &= f_l\left(\frac{\phi\left(x_i W_Q\right)^T s_i}{\phi\left(x_i W_Q\right)^T z_i} + x_i\right).
\end{aligned}$$

Linear State-Space Models

Consider a simple linear state-space model:

$$x_k = \overline{\boldsymbol{A}} x_{k-1} + \overline{\boldsymbol{B}} u_k$$
$$y_k = \overline{\boldsymbol{C}} x_k$$

- A, B, C are discretized version of learnable matrices
- Consider a single SSM layer as a replacement for an attention layer in a transformer

Linear State-Space Models

Parallelization:

$$x_k = \overline{\boldsymbol{A}} x_{k-1} + \overline{\boldsymbol{B}} u_k$$
$$y_k = \overline{\boldsymbol{C}} x_k$$

- Unroll:

$$x_0 = \overline{\boldsymbol{B}} u_0 \qquad x_1 = \overline{\boldsymbol{AB}} u_0 + \overline{\boldsymbol{B}} u_1 \qquad x_2 = \overline{\boldsymbol{A}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{AB}} u_1 + \overline{\boldsymbol{B}} u_2$$
$$y_0 = \overline{\boldsymbol{CB}} u_0 \qquad y_1 = \overline{\boldsymbol{CAB}} u_0 + \overline{\boldsymbol{CB}} u_1 \qquad y_2 = \overline{\boldsymbol{CA}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{CAB}} u_1 + \overline{\boldsymbol{CB}} u_2$$

- Reorder terms and rewrite as a convolution with kernel size L:

$$y_k = \overline{\boldsymbol{CA}}^k \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{CA}}^{k-1} \overline{\boldsymbol{B}} u_1 + \cdots + \overline{\boldsymbol{CAB}} u_{k-1} + \overline{\boldsymbol{CB}} u_k$$
$$y = \overline{\boldsymbol{K}} * u$$

$$\overline{\boldsymbol{K}} \in \mathbb{R}^L = (\overline{\boldsymbol{CB}}, \overline{\boldsymbol{CAB}}, \ldots, \overline{\boldsymbol{CA}}^{L-1} \overline{\boldsymbol{B}})$$

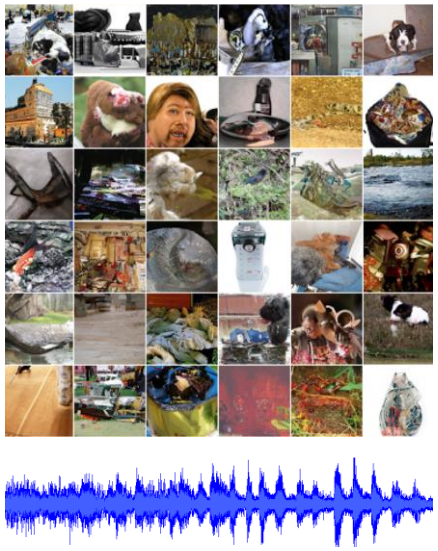
Sparse Transformer

Generating Long Sequences with Sparse Transformers

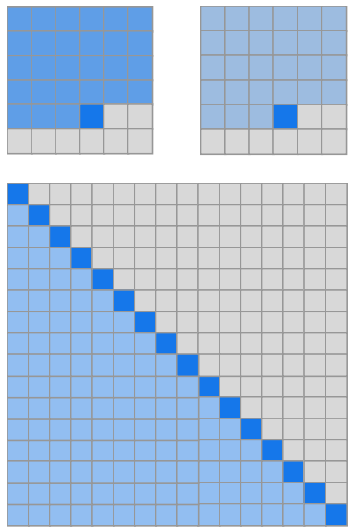
Rewon Child [1] Scott Gray [1] Alec Radford [1] Ilya Sutskever [1]

Abstract

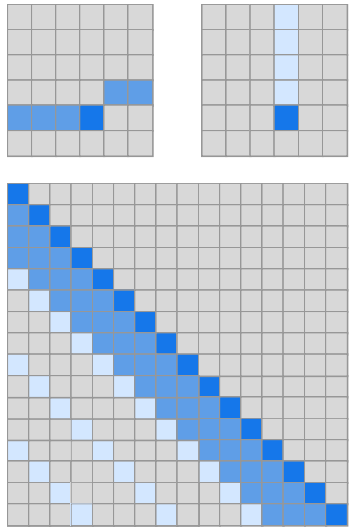
Transformers are powerful sequence models, but require time and memory that grows quadratically with the sequence length. In this paper we introduce sparse factorizations of the attention matrix which reduce this to $O(n\sqrt{n})$. We also introduce a) a variation on architecture and initialization to train deeper networks, b) the recomputation of attention matrices to save memory, and c) fast attention kernels for training. We call networks with these changes Sparse Transformers, and show they can model sequences tens of thousands of timesteps long using hundreds of layers. We use the same architecture to model images, audio, and text from raw bytes, setting a new state of the art for density modeling of Enwik8, CIFAR-10, and ImageNet-64. We generate unconditional samples that demonstrate global coherence and great diversity, and show it is possible in principle to use self-attention to model sequences of length one million or more.



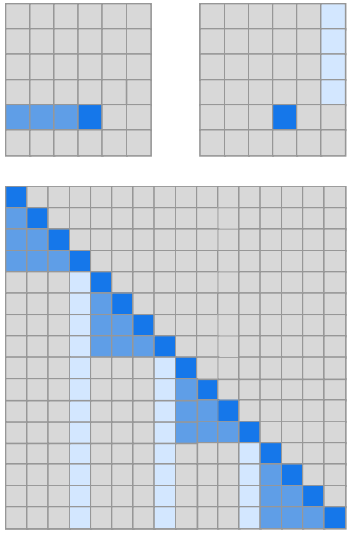
1. Introduction



(a) Transformer



(b) Sparse Transformer (strided)



(c) Sparse Transformer (fixed)

Reformer

REFORMER: THE EFFICIENT TRANSFORMER

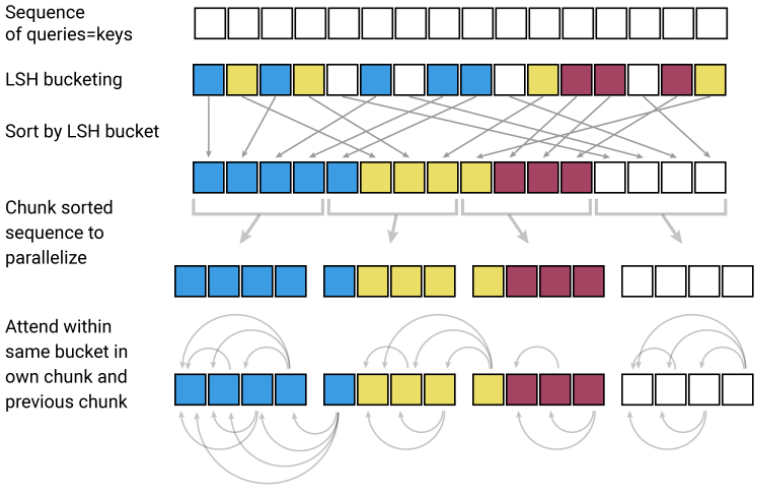
Nikita Kitaev* U.C. Berkeley & Google Research kitaev@cs.berkeley.edu

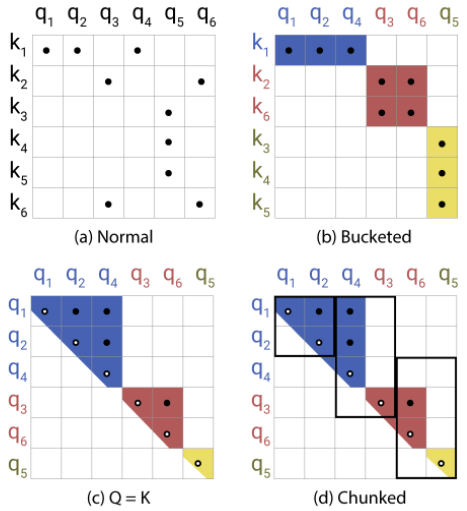
Łukasz Kaiser* Google Research {lukaszkaiser,levskaya}@google.com

Anselm Levskaya Google Research

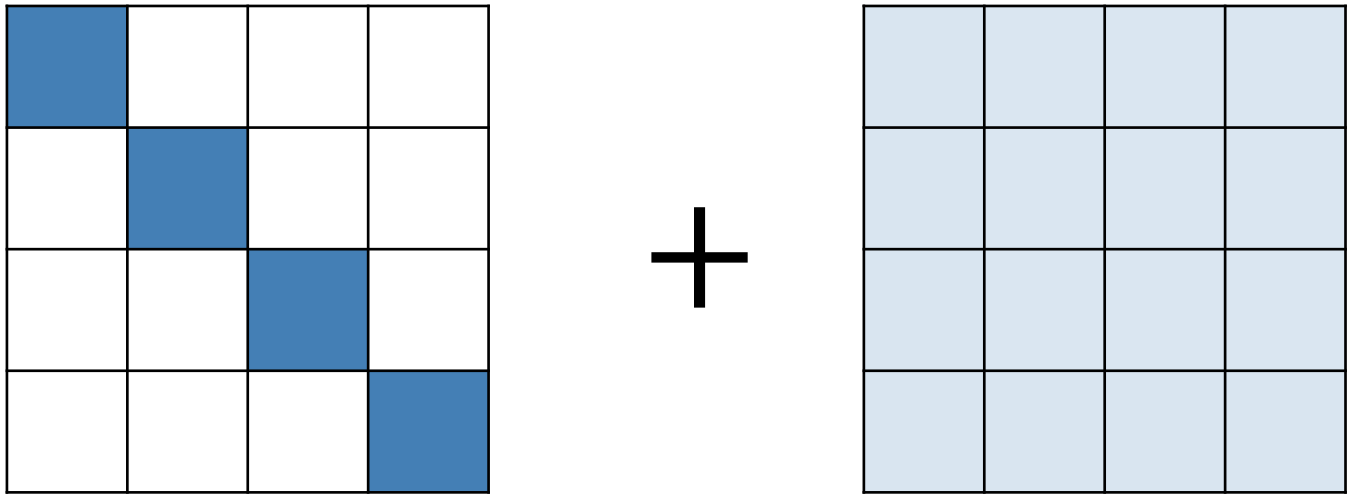
ABSTRACT

Large Transformer models routinely achieve state-of-the-art results on a number of tasks but training these models can be prohibitively costly, especially on long sequences. We introduce two techniques to improve the efficiency of Transformers. For one, we replace dot-product attention by one that uses locality-sensitive hashing, changing its complexity from O(L^2) to O($L \log L$), where L is the length of the sequence. Furthermore, we use reversible residual layers instead of the standard residuals, which allows storing activations only once in the training process instead of N times, where N is the number of layers. The resulting model, the Reformer, performs on par with Transformer models while being much more memory-efficient and much faster on long sequences.





Attention Matrix Approximation



Provably lower error!

(Informal) Approximation is unbiased and guaranteed to have lower variance.

Well-studied in stats and signal processing *(Candes et al. 09).*

Multi-Query-Attention

Fast Transformer Decoding: One Write-Head is All You Need

Noam Shazeer
Google
noam@google.com

November 7, 2019

Abstract

Multi-head attention layers, as used in the Transformer neural sequence model, are a powerful alternative to RNNs for moving information across and between sequences. While training these layers is generally fast and simple, due to parallelizability across the length of the sequence, incremental inference (where such paralleization is impossible) is often slow, due to the memory-bandwidth cost of repeatedly loading the large "keys" and "values" tensors. We propose a variant called multi-query attention, where the keys and values are shared across all of the different attention "heads", greatly reducing the size of these tensors and hence the memory bandwidth requirements of incremental decoding. We verify experimentally that the resulting models can indeed be much faster to decode, and incur only minor quality degradation from the baseline.

Positional Encoding : Rope

RoFormer: Enhanced Transformer with Rotary Position Embedding

Jianlin Su
Zhuiyi Technology Co., Ltd.
Shenzhen
bojonesu@wezhuiyi.com

Yu Lu
Zhuiyi Technology Co., Ltd.
Shenzhen
julianlu@wezhuiyi.com

Shengfeng Pan
Zhuiyi Technology Co., Ltd.
Shenzhen
nickpan@wezhuiyi.com

Ahmed Murtadha
Zhuiyi Technology Co., Ltd.
Shenzhen
mengjiayi@wezhuiyi.com

Bo Wen
Zhuiyi Technology Co., Ltd.
Shenzhen
brucewen@wezhuiyi.com

Yunfeng Liu
Zhuiyi Technology Co., Ltd.
Shenzhen
glenliu@wezhuiyi.com

November 9, 2023

Abstract

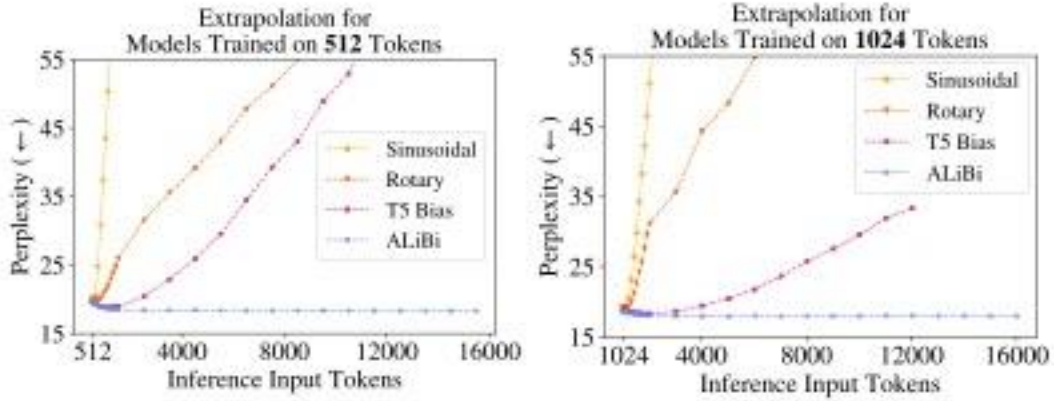
Position encoding recently has shown effective in the transformer architecture. It enables valuable supervision for dependency modeling between elements at different positions of the sequence. In this paper, we first investigate various methods to integrate positional information into the learning process of transformer-based language models. Then, we propose a novel method named Rotary Position Embedding(RoPE) to effectively leverage the positional information. Specifically, the proposed RoPE encodes the absolute position with a rotation matrix and meanwhile incorporates the explicit relative position dependency in self-attention formulation. Notably, RoPE enables valuable properties, including the flexibility of sequence length, decaying inter-token dependency with increasing relative distances, and the capability of equipping the linear self-attention with relative position encoding. Finally, we evaluate the enhanced transformer with rotary position embedding, also called RoFormer, on various long text classification benchmark datasets. Our experiments show that it consistently overcomes its alternatives. Furthermore, we provide a theoretical analysis to explain some experimental results. RoFormer is already integrated into Huggingface: https://huggingface.co/docs/transformers/model_doc/roformer

Positional Encoding: Alibi

TRAIN SHORT, TEST LONG: ATTENTION WITH LINEAR BIASES ENABLES INPUT LENGTH EXTRAPOLATION

Ofir Press[1,2] **Noah A. Smith**[1,3] **Mike Lewis**[2]

[1]Paul G. Allen School of Computer Science & Engineering, University of Washington
[2]Facebook AI Research
[3]Allen Institute for AI
ofirp@cs.washington.edu



ABSTRACT

Since the introduction of the transformer model by Vaswani et al. (2017), a fundamental question has yet to be answered: how does a model achieve extrapolation at inference time for sequences that are longer than it saw during training? We first show that extrapolation can be enabled by simply changing the position representation method, though we find that current methods do not allow for *efficient* extrapolation. We therefore introduce a simpler and more efficient position method, Attention with Linear Biases (ALiBi). ALiBi does not add positional embeddings to word embeddings; instead, it biases query-key attention scores with a penalty that is proportional to their distance. We show that this method trains a 1.3 billion parameter model on input sequences of length 1024 that extrapolates to input sequences of length 2048, achieving the same perplexity as a sinusoidal position embedding model trained on inputs of length 2048 but training 11% faster and using 11% less memory. ALiBi's inductive bias towards recency also leads it to outperform multiple strong position methods on the WikiText-103 benchmark.[1]

Transformers, BERT, and GPT-2

- A transformer uses Encoder stack to model input, and uses Decoder stack to model output (using input information from encoder side).
- If we are only interested in training a language model for the input for some other tasks, then we do not need the Decoder of the transformer, that gives us BERT.
- But if we do not have input, we just want to model the “next word”, we can get rid of the Encoder side of a transformer and output “next word” one by one. This gives us GPT.

Encoder Only Model: Bert

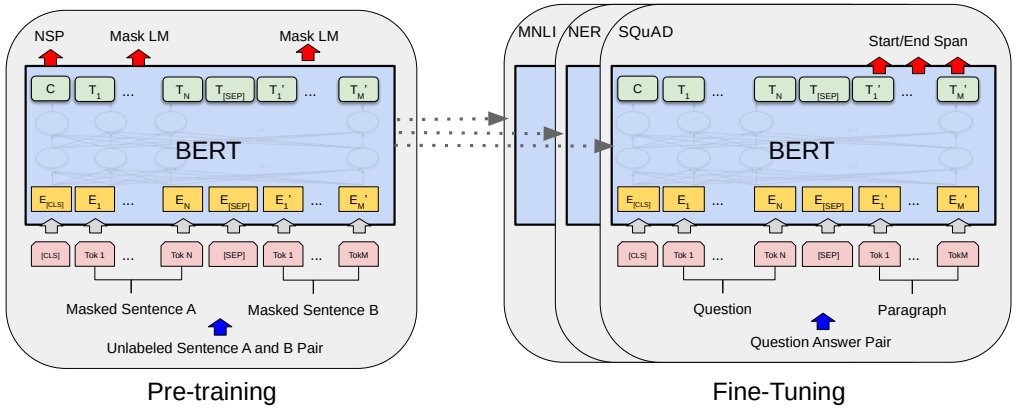
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

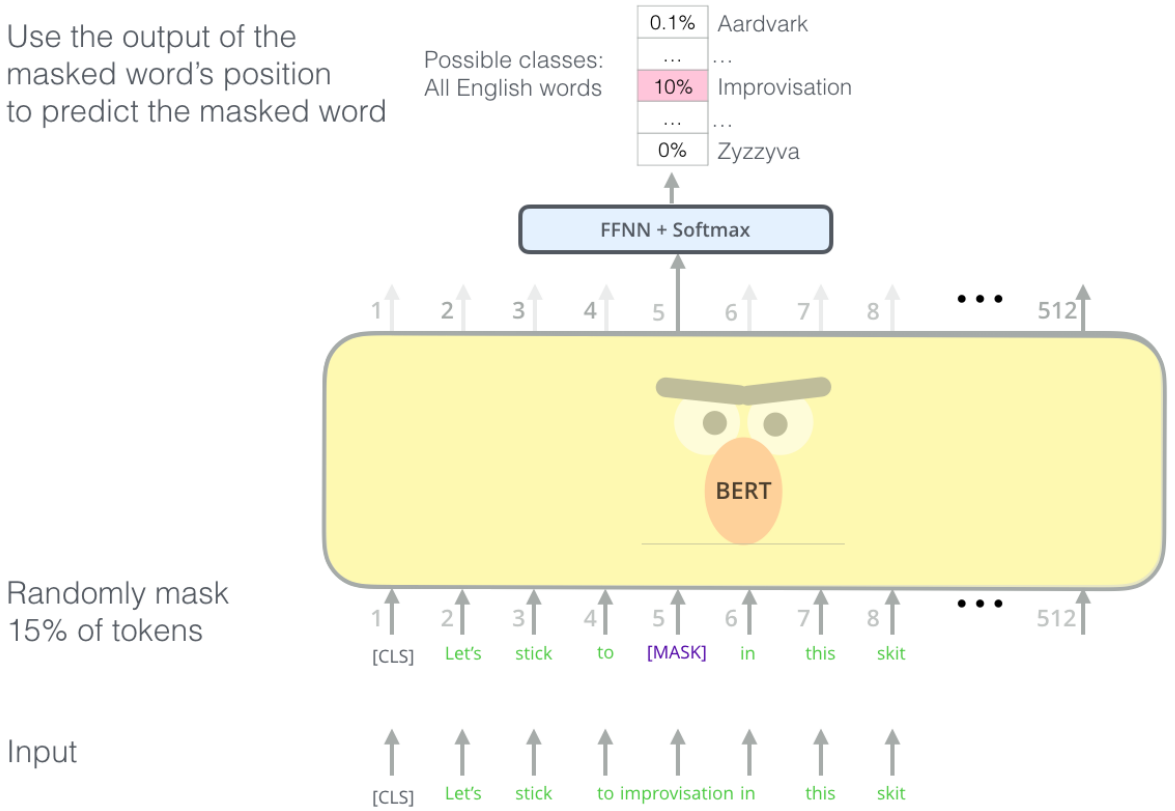
Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

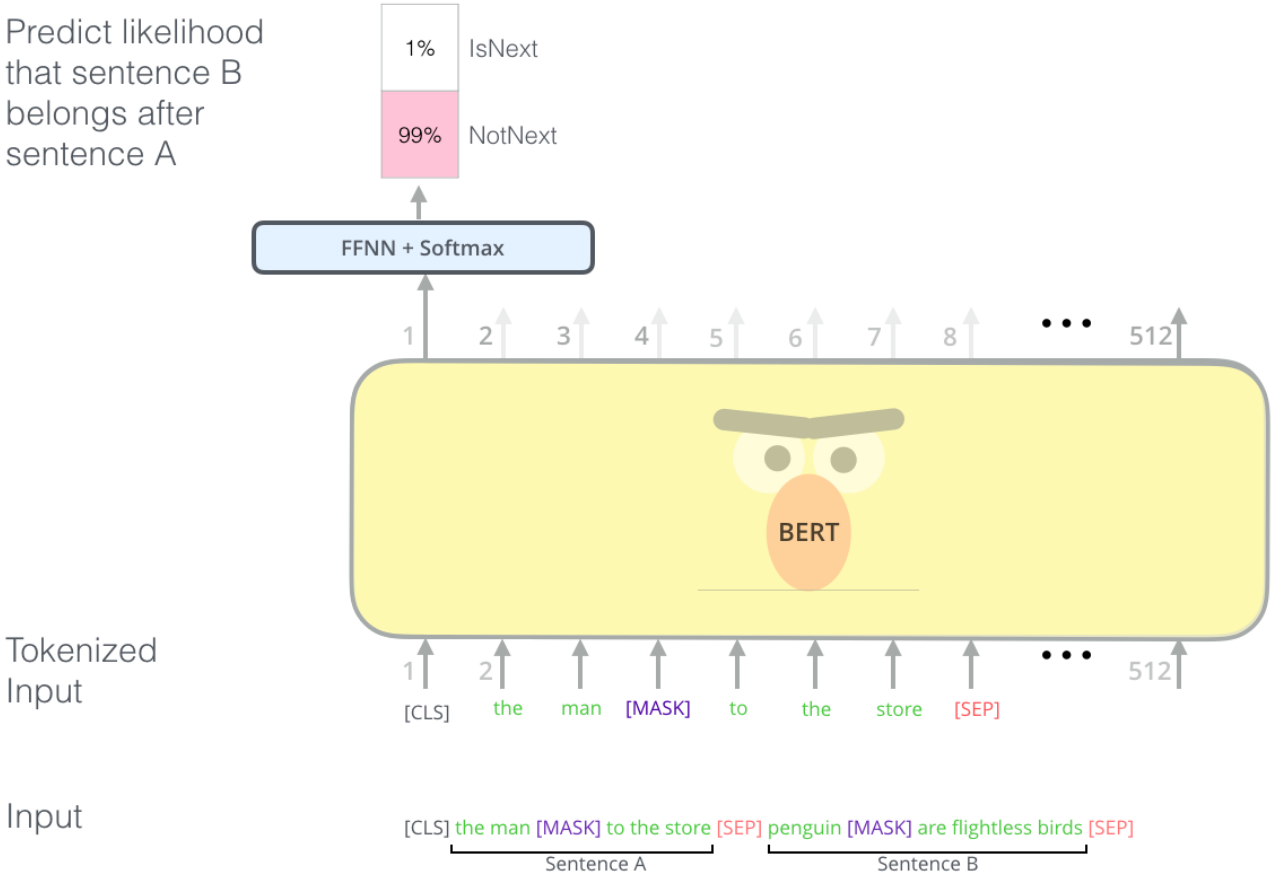


Pretraining Task 1: masked words

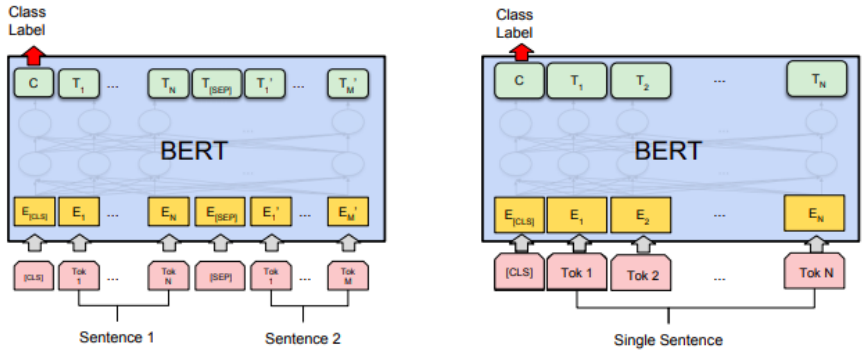


Out of this 15%,
80% are [Mask],
10% random words
10% original words

Pretraining Task 2: two sentences

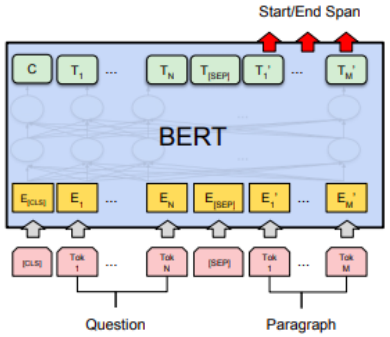


Fine-tuning BERT for other specific tasks

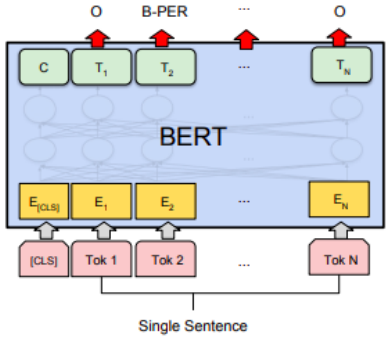


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

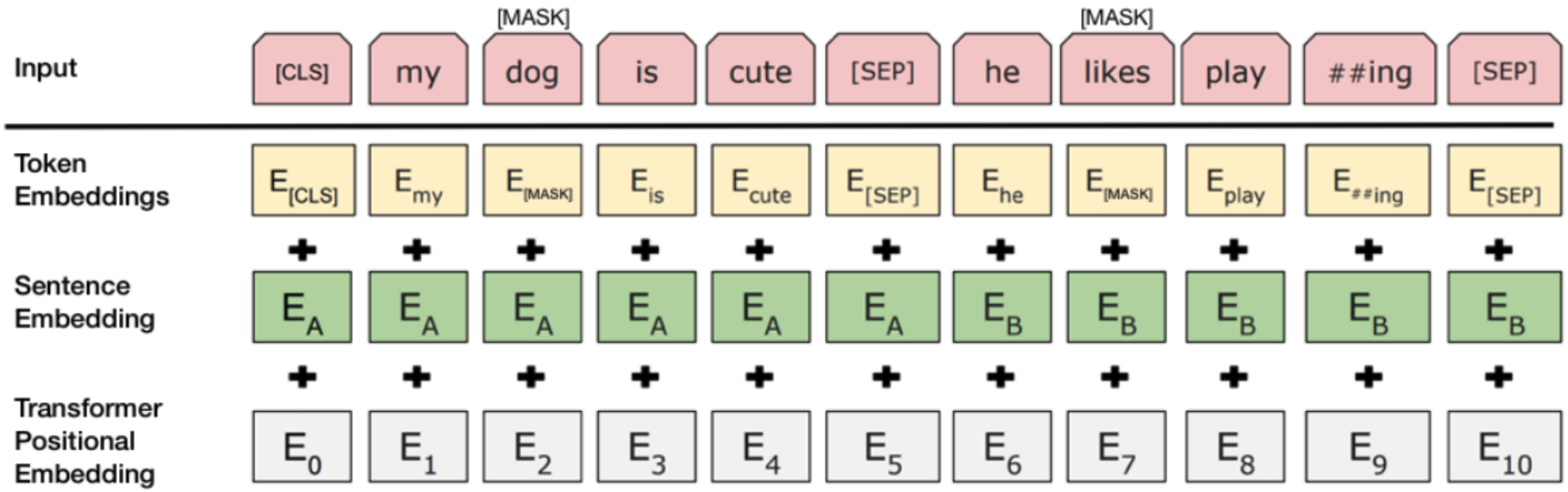


(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Pretraining Task 2: two sentences



Decoder Only Model: GPT2

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] **Jeffrey Wu** [*1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [**1] **Ilya Sutskever** [**1]

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and decaNLP (McCann et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Yogatama et al.

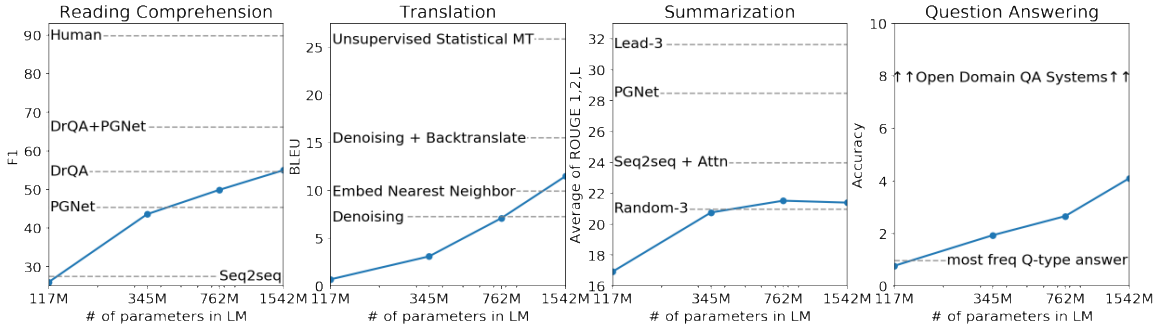
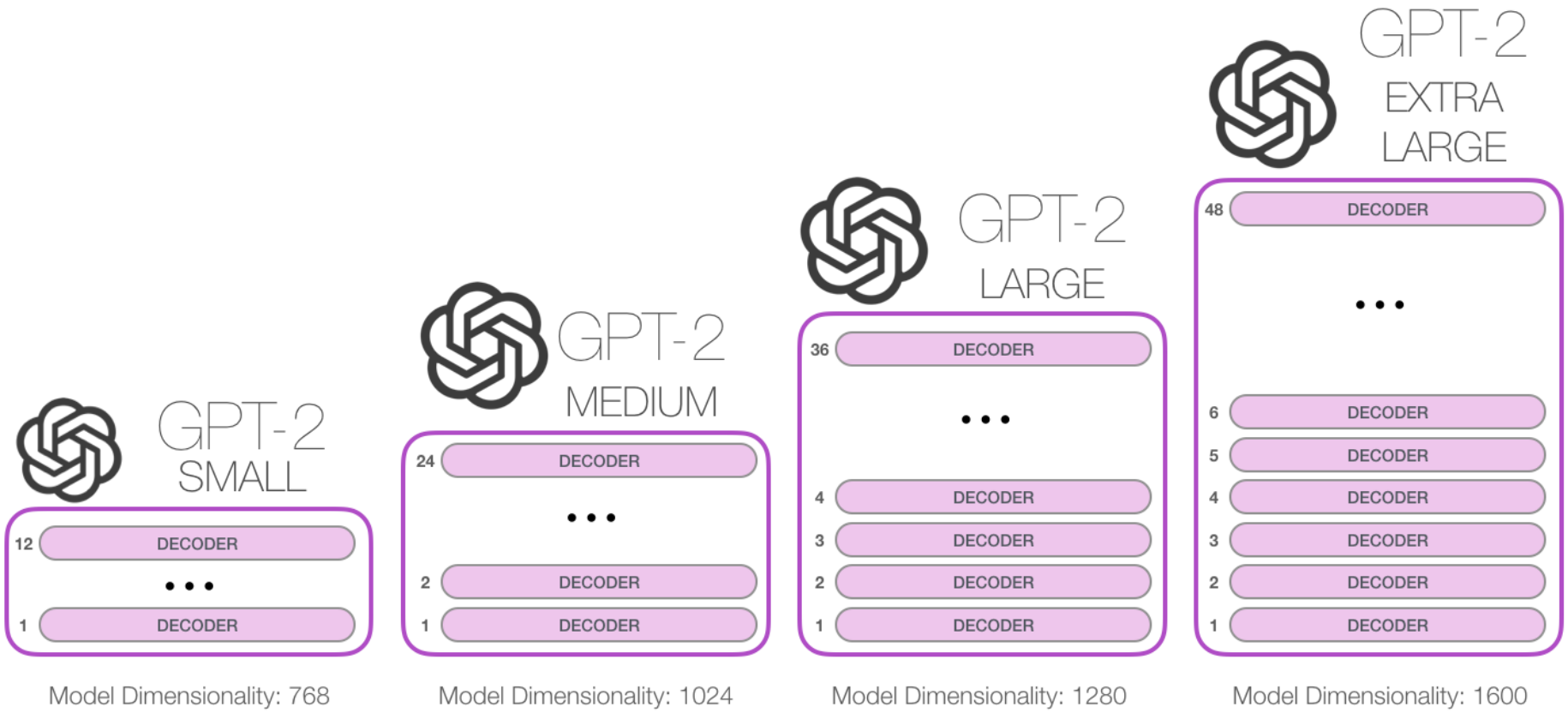


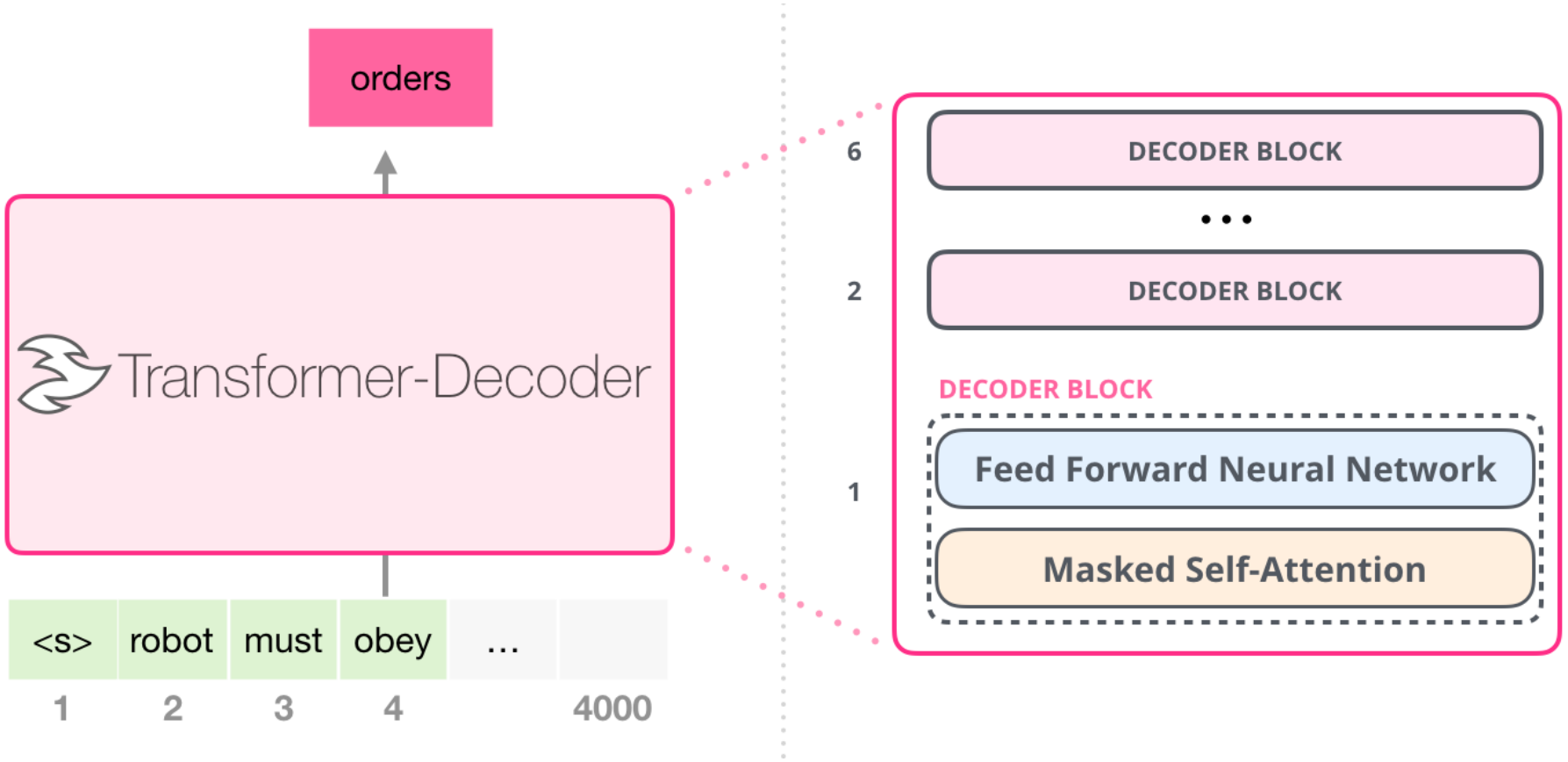
Figure 1. Zero-shot task performance of WebText LMs as a function of model size on many NLP tasks. Reading Comprehension results are on CoQA (Reddy et al., 2018), translation on WMT-14 Fr-En (Artetxe et al., 2017), summarization on CNN and Daily Mail (See et al., 2017), and Question Answering on Natural Questions (Kwiatkowski et al., 2019). Section 3 contains detailed descriptions of each result.

Scaling



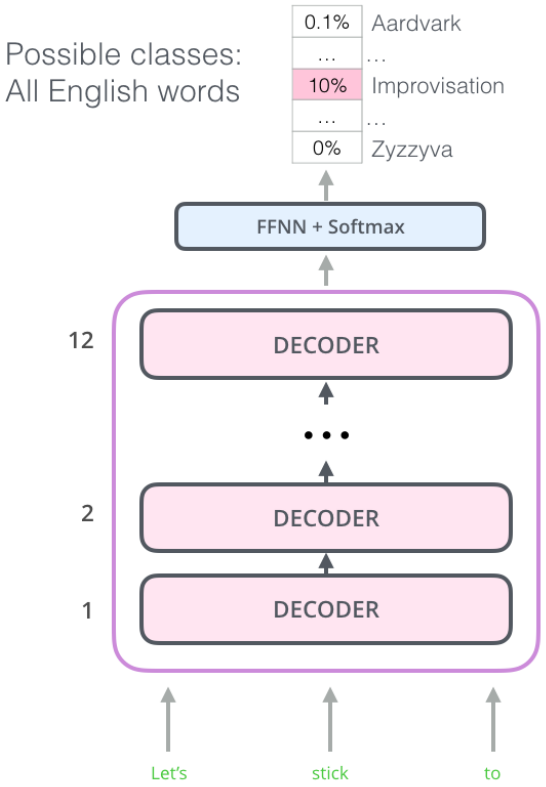
117M parameters 345M 762M 1542M

Pretraining Task: next token prediction



Re-use previous computation results: at any step, only need to results of q, k , v related to the new output word, no need to re-compute the others. Additional computation is linear, instead of quadratic.

Pretraining Data



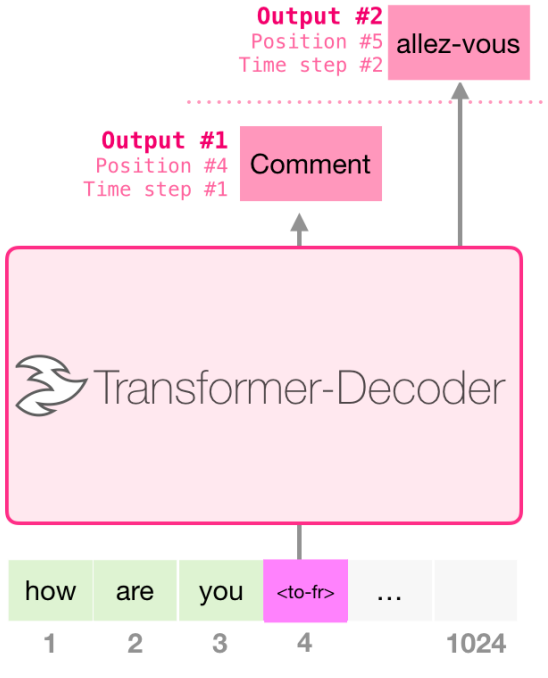
GPT-2 uses unsupervised learning approach to training the language model.

- "Our approach motivates building as large and diverse a dataset as possible in order to collect natural language demonstrations of tasks in as varied of domains and contexts as possible."
- Over 8 million documents for a total of 40 GB of text.

Downstream Tasks

Training Dataset

I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



There is no custom training for GPT-2, no separation of pre-training and fine-tuning like BERT.

- Translation
- QA
- Summarization
- Reading Comprehension
- Language Modeling

Quiz

Password: Zebra